НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

До захисту допущено
Завідувач кафедри
_____ О.В.Коваль
(підпис) (ініціали, прізвище)
“___”_____2020 р.

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

з напряму підготовки
6. 050101 “ Комп’ютерні науки’’

на тему: Система обліку надзвичайних подій природного характеру

Виконала: студент 4 курсу, групи ТМ-62

Артеменко Анастасія Олександрівна _____
(прізвище, ім’я, по батькові) (підпис)

Керівник ст. в. Шульженко Олег Феодосійович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент Головний інженер науково-інженерного центру “Волоекологія”, к.т.н. Писарук Віктор Іванович _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти перший рівень

Напрям підготовки 6.050101 “ Комп’ютерні науки’’

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ О.В. Коваль
(підпис)
”___”_____2020 р.

ЗАВДАННЯ

на дипломну роботу студенту

Артеменко Анастасії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи “ Система обліку надзвичайних ситуацій природного характеру”
керівник роботи ст. в. Шульженко Олег Феодосійович
(прізвище, ім’я, по батькові науковий ступінь, вчене звання)

затверджена наказом вищого навчального закладу від ”__”___ 202__р. № __

2. Строк подання студентом роботи ___ _____ 202__ р.
3. Вихідні дані до роботи Мова програмування C#, фреймворк Angular, середовище розробки Visual Studio/VSCode
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) проаналізувати основи обліку надзвичайних ситуацій природного характеру та розробити програмний продукт в якості системи обліку надзвичайних ситуацій, розробити інтерфейс користувача
5. Перелік ілюстраційного матеріалу архітектура бази даних, приклади роботи програмного продукту, аналітичні дані

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

Дата видачі завдання ””___ _____ 202__ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітки
1.	Затвердження теми роботи	10.10.19	
2.	Вивчення та аналіз задачі	15.01.20-6.02.20	
3.	Розробка архітектури та загальної структури системи	6.02.20-26.02.20	
4.	Розробка структур окремих підсистем	27.02.20-3.04.20	
5.	Програмна реалізація системи	4.04.20-10.05.20	
6.	Оформлення пояснювальної записки	11.05.20-02.06.20	
7.	Захист програмного продукту	10.06.20	
8.	Передзахист	10.06.20	
9.	Захист	16.06.20	

Студент _____ Артеменко А.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Шульженко О.Ф.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Метою дипломної роботи було розробити систему обліку надзвичайних ситуацій природного характеру. Програмний продукт дозволяє користувачеві переглянути інформацію про НС у вигляді точок на карті та у вигляді таблиці. На сторінці з картою є також можливість переглянути зображення з місця подій. Є можливість всі події сортувати за характером надзвичайної ситуації та за роком, коли трапилась подія. Реалізована авторизація для ідентифікації адміністратора, який має права редагування. Адміністратор, після авторизації може додавати нові події та редагувати вже існуючі.

Записка містить 32 рисунків, 2 схеми та 1 діаграму, зустрічається посилання на 11 джерел.

ABSTRACT

The purpose of the thesis was to develop an accounting system for natural nature emergencies. The software product allows the user to view information about emergencies in the form of points on the map and in the form of a table with data. On the map web-page you can also view some images from the event locale. It is possible to filter all events by the nature of the emergency and by the year when the event occurred. Implemented authorization to identify the administrator who has editing rights. After authorization administrator can add new events or edit existing ones.

The note contains 32 figures, 2 diagrams and 1 diagram, there is a link to 11 data sources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

НС – Надзвичайна ситуація;

БД – База даних;

ДСНС – Державна служба України з надзвичайних ситуацій;

CLR – Common Language Runtime, загальномовне виконуюче середовище;

CIL – Common Intermediate Language, загальна проміжна мова;

LINQ – Language Integrated Query, запити, інтегровані в мову;

HTML – HyperText Markup Language, мова розмітки гіпертексту;

CSS – Cascading Style Sheets, каскадні таблиці стилів;

JWT – JSON Web Token, це стандарт токена доступу на основі JSON;

HMAC – Hash-based message authentication code, хеш-код аутентифікації повідомлень;

RSA – абревіатура від прізвищ Rivest, Shamir та Adleman, криптографічний алгоритм з відкритим ключем;

ECDSA – Elliptic Curve Digital Signature Algorithm, алгоритм з відкритим ключем для створення цифрового підпису;

SOLID — це абревіатура складена з перших літер п'яти базових принципів об'єктно-орієнтованого програмування та дизайну запропонована Робертом Мартіном;

HTTP — протокол передачі даних;

API - Application Programming Interface, прикладний програмний інтерфейс.

ВСТУП

Останні декілька десятиліть інформаційні технології розвиваються дуже стрімко.

Особливих змін зазнали системи збору, аналізу, зберігання даних. Дуже мало існує інформації, яку було б неможливо знайти в цифровому форматі. До того, як виникли системи збору та аналізу даних, процес документування певної надзвичайної ситуації був важким. Щоб дізнатися деталі події потрібно було витратити багато часу, обійти всіх очевидців, записати їх показання та зберегти. В той самий час, аналіз та збір статистичних даних потребував детального вивчення всіх матеріалів, тож займав не один день. У випадку коли необхідно отримати конкретну інформацію з будь-якої сфери, необхідно обійти всі відповідні архіви.

Сучасні технології дозволять зберігати терабайти інформації за допомогою баз даних в інформаційних системах, більш того, їх обробка займає лічені секунди. Маючи такі потужні інструменти бажано застосовувати їх в якомога більшій кількості сферах діяльності. Наприклад, задля збору та зберігання інформації про надзвичайні ситуації, точніше надзвичайні ситуації природного характеру та їх статистику.

Людство від самого свого початку намагалося дослідити навколишнє середовище. Століття спостережень дали можливість передбачати деякі ситуації та поведінку природних процесів. Стало можливим зрозуміти місце та, що важливіше характер надзвичайних ситуацій, таких як, землетрус, повінь, буря, ураган і т.д..

В сучасному світі є спеціальні організації, що займаються дослідженням змін у навколишньому середовищі, тож про більшість надзвичайних ситуацій є можливість дізнатися заздалегідь, інколи навіть за роки.

На жаль, як і сотні років тому, людство може тільки дізнаватися та не має змоги боротися з силами природи, тож не може зупинити надзвичайні ситуації природного характеру.

Навіть більше, НС природного характеру одні з найнебезпечніших. Наприклад, одна пожежа може розповсюдитись настільки сильно, що для боротьби з вогнем знадобиться не один тиждень.

Нещодавно на території України сталась масштабна пожежа в Чорнобильській зоні. Площа пожежі склала понад десять тисяч гектарів, чим нанесла великої шкоди флорі та фауні. Тисячі гектарів вкрило чорним димом, через що сотні тварин постраждали, хтось безпосередньо від вогню, інші від нестачі їжі, оскільки більшість трави та соломи вигоріло, решта через втрату свого будинку. Дим від пожежі розповсюдився на сотні кілометрів, а задля того щоб потушити всю територію знадобилось понад 500 чоловік та понад 120 одиниці техніки [1].

Однак, можна відстежувати НС які вже сталися, та аналізувати їх походження. За допомогою цих знань можна точніше визначати місце наступного НС, чим максимально скоротити кількість жертв та збитки на відновлення.

Для такого роду досліджень було б досить ефективно також спостерігати та контролювати кількість постраждалих, збитки нанесені попередніми надзвичайними ситуаціями та вартість їх ліквідації за регіонами та роками. Наприклад, якщо визначити в яких регіонах найчастіше виникають пожежі за останні роки, то можна в цих місцевостях вжити запобіжні міри.

Відкривши зведення новин з приводу повеней в Україні за останні сто років, то одразу впадає в очі те, що переважна більшість надзвичайних ситуацій, викликаних великою кількістю води, зустрічається на Закарпатті. Це досить очевидно, якщо брати до уваги Карпатські гори, що багаті на опади. Однак коли ця статистика показана більш наочним чином, на карті, то одразу стає зрозумілим необхідність докласти допоміжних зусиль для запобігання великих жертв повені. Такого роду наочність корисна не лише для повені, а й для інших типів надзвичайних ситуацій природного характеру.

1. ПОСТАНОВКА ЗАДАЧІ СТВОРЕННЯ СИСТЕМИ ОБЛІКУ НАДЗВИЧАЙНИХ СИТУАЦІЙ ПРИРОДНОГО ХАРАКТЕРУ

Надзвичайні ситуації природного характеру завжди відстежувались людиною та в певному форматі зберігались. З плином часу вид зберігання даних удосконалювався та ставав все зручнішим для використання. В наш час одним з найзручніших варіантів зберігання даних, таких як інформація про надзвичайну ситуацію є система обліку.

Розроблений програмний продукт має бути цілісною системою обліку. Це означає, що повинна бути можливість збирати, обробляти та в певному форматі віддавати користувачеві інформацію.

Було вирішено розділити задачу створення системи на наступні підпункти:

1.1 Задача додавання нової НС.

Оскільки надзвичайні ситуації природного характеру неможливо припинити чи будь-яким чином запобігти, має бути можливість додавання нової події до системи;

1.2 Задача відображення події на карті

Для більш наглядного відображення інформації було обрано відображення на кожної події на карті з можливістю переглянути деталі кожної надзвичайної ситуації;

1.3 Задача відображення події в таблиці

Оскільки подія відображена на карті наглядна, але не сильно інформативна, повинна бути можливість переглянути всі доступні події у форматі таблиці з даними, щодо НС;

1.4 Задача відображення статистики

Для попереднього аналізу ситуації потрібно показати користувачеві кількість НС з початку року, та якої шкоди вони заподіяли;

1.5 Задача фільтрації НС за роками

Подій на карті та в таблиці може виявитися більше ніж можливо опрацювати візуально. Тож, для зручного використання програмного продукту потрібно надати користувачеві можливість відфільтрувати надзвичайні ситуації за роками їх виникнення;

1.6 Задача авторизації

Задля того, щоб будь-який користувач не мав можливості додати нереальну подію, або ввести некоректні дані, потрібно додати можливість авторизації. Тільки авторизований адміністратор має можливість додати інформацію стосовно перевіреної події.

2. АНАЛІЗ ПРОБЛЕМИ СИСТЕМ ОБЛІКУ НАДЗВИЧАЙНИХ СИТУАЦІЙ

Більшість українців щодня переглядають хоча б одне зведення новин. Зазвичай згадують найбільш вагомі та вражаючі новини. Людина в свою чергу запам’ятовує тільки деякі з них, вибірково. Дуже важко втримати в голові новини одного характеру за довгий час.

Оскільки природна сила дуже вагома та може завдати великих збитків, про надзвичайні ситуації природного характеру зазвичай говорять голосно та довго.

В Україні найчастіше спостерігаються такі надзвичайні ситуації природного походження:

- небезпечні геологічні явища: зсуви, обвали, просадки земної поверхні різного походження;
- небезпечні метеорологічні явища: зливи, урагани, сильні снігопади, сильний град, ожеледь;
- небезпечні гідрологічні явища: повені, паводки, підвищення рівня ґрунтових вод;
- природні пожежі лісових та торф'яних масивів;
- масові інфекції та хвороби людей, тварин, рослин [2].

Про всі НС можна дізнатися з новин чи зі звітів розміщених на сайті державної служби України з надзвичайних ситуацій.

Державна служба України з надзвичайних ситуацій (ДСНС) є центральним органом виконавчої влади, діяльність якого спрямовується і координується Кабінетом Міністрів України через Міністра внутрішніх справ і який реалізує державну політику у сфері цивільного захисту, захисту населення і територій від надзвичайних ситуацій та запобігання їх виникненню, ліквідації наслідків надзвичайних ситуацій, рятувальної справи, гасіння пожеж, пожежної та техногенної безпеки, діяльності аварійно-рятувальних служб, а також гідрометеорологічної діяльності.

Основними завданнями ДСНС є:

1) реалізація державної політики у сфері цивільного захисту, захисту населення і територій від надзвичайних ситуацій, запобігання їх виникненню, ліквідації наслідків надзвичайних ситуацій, рятувальної справи, гасіння пожеж, пожежної та техногенної безпеки, діяльності аварійно-рятувальних служб, а також гідрометеорологічної діяльності;

2) здійснення державного нагляду (контролю) за додержанням і виконанням вимог законодавства у сфері цивільного захисту, пожежної та техногенної безпеки, діяльності аварійно-рятувальних служб;

3) внесення на розгляд Міністра внутрішніх справ пропозицій щодо забезпечення формування державної політики у зазначених сферах;

4) реалізація в межах повноважень, передбачених законом, державної політики у сфері волонтерської діяльності [3].

Дані про кожну НС зберігаються у архівах новинних каналів чи ДСНС. Зазвичай створюються довідники з статистикою надзвичайних ситуацій на сайті ДСНС за певний період часу. В того роду довідниках чи звітах знаходяться порівняльні таблиці за різними критеріями.

Наприклад як зображено нижче:

Дані про надзвичайні ситуації	2018 рік	2019 рік	Зменшення (збільшення), у відсотках
Загальна кількість НС:	**128**	**146**	**14,1** ⬆
В тому числі:			
Техногенного характеру	48	60	**25,0** ⬆
Природного характеру	77	81	**5,2** ⬆
Соціального характеру	3	5	**66,7** ⬆
В тому числі за рівнями:			
Державного рівня	2	2	**0,0**
Регіонального рівня	6	7	**16,7** ⬆
Місцевого рівня	64	63	**1,6** ⬇
Об'єктового рівня	56	74	**32,1** ⬆
Загинуло людей внаслідок НС	**168**	**199**	**18,5** ⬆
Постраждало людей внаслідок НС	**839**	**1492**	**77,8** ⬆
Матеріальні збитки від НС, тис. грн.	**496 965**	**685 269**	**37,9** ⬆

Рисунок 1. Кількісні показники НС, що виникли у 2019 році, порівняно із 2018 роком

Вид НС	Кількість НС		Загинуло людей		Постраждало людей	
	2018 р.	2019 р.	2018 р.	2019 р.	2018 р.	2019 р.
НС техногенного характеру						
НС унаслідок аварій чи катастроф на транспорті	18	16	63	75	80	47
НС унаслідок пожеж, вибухів	22	27	52	79	9	81
у тому числі у будівлях або спорудах житлової призначеності	*17*	*13*	*50*	*48*	*2*	*17*
НС унаслідок наявності у навколишньому середовищі шкідливих і радіоактивних речовин понад ГДК	2	3	0	0	0	0
НС унаслідок раптового руйнування будівель і споруд	0	4	0	10	0	14
НС унаслідок аварій в електроенергетичних системах	1	0	0	0	0	0
НС унаслідок аварій у системах життєзабезпечення	5	10	0	0	0	0
Всього НС техногенного характеру	48	60	115	164	89	142
НС природного характеру						
Геологічні НС	1	0	0	0	0	0
Метеорологічні НС	4	16	0	6	0	13
Гідрологічні НС поверхневих вод	2	0	0		0	0
НС, пов'язані з пожежами у природних екологічних системах	9	8	0	0	2	0
Медико-біологічні НС	61	57	47	15	744	1334
НС природного характеру	77	81	47	21	746	1347
НС соціального характеру						
Встановлення вибухового пристрою у багатолюдному місці, установі (організації, підприємстві), житловому секторі, транспорті	1	1	0	1	4	2
НС, пов'язані з нещасними випадками з людьми	2	4	6	13	0	1
Всього НС соціального характеру	3	5	6	14	4	3
Всього НС	**128**	**146**	**168**	**199**	**839**	**1492**

Рисунок 2. Статистичні дані щодо кількісних показників класифікованих НС

№ з/п	Регіон України	2010 рік	2011 рік	2012 рік	2013 рік	2014 рік	2015 рік	2016 рік	2017 рік	2018 рік	2019 рік	Всього НС за 10 років
1	Авт.Республіка Крим	12	4	18	3	1						38
2	Вінницька	9	7	6	3	6	5	4	7	3	4	54
3	Волинська	4	7	1	8	6	6	10	5	6	7	60
4	Дніпропетровська	17	10	5	3	6	5	8	4	9	9	76
5	Донецька	35	29	25	20	20	7	6	24	10	10	186
6	Житомирська	8	6	10	3	5	5	8	5	5	8	63
7	Закарпатська	9	5	5	7	4	5	6	9	8	7	65
8	Запорізька	10	11	8	4	7	5	1	3	4	3	56
9	Івано-Франківська	8	4	3	3	2	3	5	7	2	7	44
10	Київська	9	7	13	4	3	13	5	6	6	7	73
11	Кіровоградська	3	3	3	2	2	2	6	3	3	4	31
12	Луганська	26	18	13	11	7	3	4	5	9	7	103
13	Львівська	11	14	20	14	7	4	6	3	5	8	92
14	Миколаївська	9	12	11	7	7	7	10	8	8	5	84
15	Одеська	14	17	13	10	9	7	14	13	5	11	113
16	Полтавська	3	2	3	5	5	8	10	5	4	10	55
17	Рівненська	5	11	4	3	5	4	6	10	2	5	55
18	Сумська	5	4	5	6	4	7	7	3	5	5	51
19	Тернопільська	3	9	8	3	7	5	2	3	6	4	50
20	Харківська	14	11	11	8	10	9	6	10	4	3	86
21	Херсонська	19	6	15	4	10	4	5	9	7	4	83
22	Хмельницька	11	6	6	4	5	4	4	2	3	2	47
23	Черкаська	4	4	9	4	2	5	3	9	5	1	46
24	Чернівецька	11	1	2	2	1	4	7	5	2	5	40
25	Чернігівська	7	7	6	5	8	9	8	3	5	5	63
26	м. Київ	4	4	9	8	3	14	7	12	7	5	73
27	м. Севастополь	3	1	1	1	1						7
28	За межами України	0	1	0	0	0	0	0	0	0	0	1
Всього НС*		**254**	**221**	**212**	**144**	**143**	**148**	**149**	**166**	**128**	**146**	**1711**
В тому числі:												
техногенного характеру		135	134	120	76	74	63	56	50	48	60	816
природного характеру		108	77	74	56	59	77	89	107	77	81	805
соціального характеру		11	10	18	12	10	8	4	9	3	5	90
В тому числі:												
Державного рівня		5	4	1	1	5	2	1	2	2	2	25
Регіонального рівня		16	3	13	12	9	9	9	8	6	7	92
Місцевого рівня		107	89	83	58	59	62	64	70	64	63	719
Об'єктового рівня		126	125	115	73	70	75	75	86	56	74	875
Загинуло людей		**361**	**355**	**301**	**253**	**287**	**242**	**183**	**172**	**168**	**199**	**2521**
Постраждало людей		**753**	**985**	**861**	**854**	**680**	**962**	**1805**	**892**	**839**	**1492**	**10123**
Матеріальні збитки, млн.		**984,70**	**102,75**	**249,79**	**396,33**	**198,85**	**532,72**	**265,31**	**896,80**	**496,97**	**685,27**	**4809,49**

Рисунок 3. Кількісні показники класифікованих НС, які сталися на території України у 2010-2019 роках

Однак, переглянути конкретну подію неможливо, потрібно шукати її власноруч, що не дуже зручно.

Існує така тенденція, що в одному і тому ж місці виникають НС одного типу.

Наприклад повені в певний період часу (зазвичай на весні, коли тане сніг) наступають в одних і тих же місцях або схожих за місцевістю.

Якби була можливість переглянути за картою в яких місцях були повені то стало більш зрозуміло їх природу та характер. Маючи інформацію про НС у вигляді карти більш наглядно можна зрозуміти які регіони та місця потребують підтримки та допомоги. Є можливість проаналізувати ситуацію

та докласти максимальних зусиль задля того, щоб збитки нанесені повінню були мінімальними, адже на сьогодні збитки спричиненні цим явищем є найбільшими серед всіх надзвичайних ситуацій природного характеру та складають близько 40%.

Аналіз надзвичайних ситуацій, що стались раніше, за допомогою карти є дійсно необхідним інструментом для мінімізації шкоди заподіяної будь яким явищем.

3. Засоби розробки

Для створення системи обліку надзвичайних ситуацій природного характеру було обрано 2 основні мови програмування C# для частини backend та Angular для частини frontend. Обидві мови часто використовувані, та досить легко поєднуються в одному проекті. Для зручності розробки програмного продукту було використано Microsoft Visual Studio та VSCode в якості середовища розробки.

Для глобального доступу до розробленого програмного продукту, його було завантажено в глобальну мережу Internet за допомогою Microsoft Azure Cloud. Для зберігання даних було обрано Microsoft Azure SQL Database. Реалізація авторизації втілена за допомогою jwt authentication.

Далі про кожну технологію детальніше.

3.1 Backend

.Net Core це безкоштовна та відкрита платформа для розробки програмного забезпечення. Він працює з такими операційними системами як Linux, MacOs та Windows. .Net Core основана на .Net Framework, відмінність в тому що Core кросплатформений та модульний. Також не вагомим плюсом є можливість роботи з хмарними технологіями.

Головні особливості .Net Core:

- Кросплатформеність;
- Відкритий код;
- Сучасні технології;
- Висока продуктивність;
- Можливість однакового виконання коду на різних операційних системах;
- Зручні засоби для роботи з командним рядком;
- Гнучкість розробки, тобто є можливість інтеграції в різні системи та використання з контейнером Docker [4].

C# це об’єктно орієнтовна мова програмування. Вважається одним з нащадків мови С, тож має синтаксис схожий на інші с-подібні мови. Від початку була створена як мова для розробки програмного продукту для платформи .Net Framework.

Незважаючи на те, що C# вважається об’єктно-орієнтованою мовою, вона додатково включає підтримку компонентно-орієнтованого програмування. Головною перевагою компонентів в розробці є представлення моделі програмування з її властивостями, методами та подіями. Вони мають атрибути, які надають декларативну інформацію про компонент та мають власну документацію.

Довговічність програмних продуктів, створених на основі мови C# також забезпечує така функція, як збір сміття (Garbage collection). Ця функція притаманна далеко не всім популярним мовам програмування. Суть закладається в тому, що автоматично, без виклику очищення, відновлюється пам’ять, яка зайнята недоступними більше в програмі об’єктами. Це зручно тому що кількість зайнятої пам’яті знижується до мінімуму.

C# на відміну від більшості мов програмування приділяє увагу розбиттю програм та бібліотек на версії. Завдяки цьому зі зміною версій програми, змінюється і версія використовуваних бібліотек, що робить ризик поломки дуже малий [5].

Особливістю мови C# вважають те що вона розроблена як мова прикладного рівня та має працювати напряму з CLR та напряму залежить від можливостей CLR.

CLR це Common Language Runtime, тобто загальномовне виконуюче середовище та вважається одним із основних компонентів пакета Microsoft .NET Framework. CLR це середовище для байткода CIL, в якому компілюються програми написані на мовах сумісних з .Net Framework, в даному випадку з C#.

Також в розробці було використано Entity Framework Core. Це кросплатформена версія технології доступу до даних. Вона дозволяє

розробникам на .Net працювати з базами даних через об’єкти, що значно спрощує роботи та скорочує кількість коду, необхідного для доступу до баз даних.

Весь код, що потребується для доступу до БД замінюється на моделі об’єктів, що зберігаються. Ці моделі складаються з класів, що забезпечують взаємодію з БД та дозволяють отримувати та записувати дані. За допомогою Entity Framework Core можна створити модель з існуючої бази даних, або навпаки, створити БД з описаної моделі даних.

Основні поняття для Entity Framework Core:

- Модель

За допомогою EF Core доступ до даних здійснюється за допомогою моделі. Модель складається з класів сутностей та об'єкта контексту, який представляє сеанс із базою даних, що дозволяє запитувати та зберігати дані.

- Запити

Екземпляр класу моделі витягуються з бази даних за допомогою запитів Language Integrated Query(LINQ).

- Зберігання даних

Створення, видалення та редагування даних виконується за допомогою екземпляру класу моделі даних [10].

3.2 База даних

Для управління базою даних було обрано Microsoft SQL Server. Це система управління реляційними базами даних від компанії Microsoft. Існують багато різновидів Microsoft SQL Server спрямованих на різні сегменти ринку, він невеликих одномашиних додатків, до ентерпрайзних проектів з мільйонами клієнтів та їхньою інформацією. Завдяки великому досвіду роботи з базами даних є можливість керувати середовищем великих даних за допомогою кластерів. В систему SQL Server глибоко інтегровані інструменти для аналітики даних, що повністю підтримуються службами Microsoft [6].

В розробленому програмному продукті базу даних було розміщено в хмарному середовищі. Для цього використано SQL Server on Azure. Azure це

сервер хмарних обчислень, розроблений компанією Microsoft. Основними задачами Azure є управління, побудова, розгортання та тестування додатків та послуг. Даний сервер підтримує безліч мов програмування, фреймворків та інструментів не тільки створених компанією Microsoft, а й сторонніми [7].

3.3 Fronend

Перед початком розробки інтерфейсу користувача концепт було розроблено та зображено за допомогою онлайн сервісу розробки інтерфейсів Figma.

Це система підходить для створення прототипів та дизайн систем. Головною особливістю є хмарна технологія, яка відкриває можливості для розробників та інших членів команди беруть участь в розробці продукту.

Переваги Figma:

- Безкоштовність;
- Режим роботи з декількома користувачами на одній робочій зоні;
- Зберігання файлів;
- Збереження попереднього перегляду зображень;
- Можливість залишати коментар відразу на макетах;
- Режим презентацій для оцінки вигляду на повному екрані;
- Висока продуктивність;
- Можливість перенести налаштовані елементи в CSS код;
- Легкість редагування за рахунок сітки та автоматичного розміщення;
- Можливість додавання безлічі макетів на одну робочу поверхню[11].

Відтворення екрану користувача було вирішено розробити за допомогою фреймворку Angular. Angular - це структура дизайну прикладних програм та платформа розробки для створення ефективних та складних одно сторінкових додатків.

Серед переваг можна відзначити:

- Кросплатформеність

Є можливість використовувати сучасні можливості веб платформи. Висока продуктивність та можливість почату розробку з нульового кроку, без багатьох допоміжних установок. Можливість створювати native мобільні додатки. Можливість створювати десктопні додатки на Mac, Windows та Linux, так само легко як і веб додатки, за допомогою тих самих інструментів та методів.

- Швидкість

Швидкість забезпечується автоматичною генерацією програмного коду. Angular додаток складається з багатьох окремих компонентів, для кожного з яких є шаблони. Однак, Angular перетворює їх в код та оптимізує. Тож кінцевий код який перевівся в JS більш оптимізований ніж програмний код на JavaScript.

- Універсальний

Може легко працювати на Node.js®, .NET, PHP та інших серверах для майже миттєвого візуалізації лише в HTML і CSS.

- Продуктивність

Швидко створюйте представлення інтерфейсу користувача з простим та потужним синтаксисом шаблону. Значно допомагає при розробці використання CLI. Завдяки інструментам командного рядку можна миттєво створити компоненти, сервіси, модулі, почати тестування чи запустити зборку проекту для розгортання.

- Надійність

Завдяки типізації є можливість запобігти багато помилок, що могли б не виявитись від початку. В Angular більшість помилок відгладжуються на етапі зборки [8].

3.4 Авторизація

В розробленому програмному продукті використовувалась JWT автентифікація. Її суть в тому, що після того, як користувач успішно ввійшов в систему, програма створює JWT та відправляє її назад користувачеві. Після чого всі наступні запити користувача будуть містити в собі JWT.

Відсутність JWT в запиті при відправці на backend означає що користувач не авторизований, тож інформації він не отримає, його має перенести на сторінку авторизації.

JSON Web Token (JWT) - це відкритий стандарт (RFC 7519), який визначає компактний та автономний спосіб безпечної передачі інформації між сторонами як об'єкт JSON. JWT можуть бути підписані за допомогою секретного (за допомогою алгоритму HMAC) або пари відкритого / приватного ключів за допомогою RSA або ECDSA [9].

3.5 Хмарне середовище

Хмарне обчислення це набір служ таких як сервери, сховища даних, бази даних, програмне забезпечення та інше.

Переваги:

- Вартість

Не потребую початкового вкладення коштів на обладнання та програмне забезпечення.

- Швидкість

Більшість сервісів хмарних обчислень надаються самообслуговуванням та за потребою, тому навіть величезні обсяги обчислювальних ресурсів можна забезпечити за лічені хвилини, як правило, лише декількома клацаннями миші, що надає бізнесу велику гнучкість та знімає планування потужності.

- Глобальне масштабування

Має можливість еластичного масштабування, тобто використовується тільки та кількість ресурсів, що справді потрібна. Якщо не вистачає обчислювальної потужності, то можна з легкістю доставити потрібну кількість ресурсів.

- Продуктивність

Найбільші послуги хмарних обчислень працюють у всесвітній мережі захищених центрів обробки даних, які регулярно оновлюються до найновішого покоління швидкого та ефективного обчислювального обладнання. Це дає ряд переваг у порівнянні з одним корпоративним центром

обробки даних, включаючи скорочення мережевої затримки для додатків та більшу економію масштабу.

- Надійність

Під час використання хмарних сервісів, робляться резервні копії, тож ймовірність втратити свої дані під час будь-якого фізичного збою малоймовірна.

4.ОПИС ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Для створення програмного продукту у вигляді системи обліку надзвичайних подій природного характеру необхідно реалізувати наступні частини:

- Створення та хостинг бази даних
- Реалізація серверної частини
- Реалізація клієнтської частини
- Налагодження зв’язку між сервером та клієнтом
- Робота з авторизацією

Тепер розберемо детальніше кожен з пунктів:

4.1 Створення та хостинг бази даних

В якості бази даних було визначено використовувати SQL Server, та для зручності розмістити базу даних у хмарі Azure.

Для доступу та роботи з базою даних використано Entity Framework Core. Головною концепцією Entity Framework є поняття об’єкта, який описує набір даних, що зв’язані з певній об’єктом. Саме через це дана технологія працює з об’єктами та їх колекціями, а не з таблицями. Кожен об’єкт має набір характеристик, що відповідають полям таблиці. Об’єкти пов’язані один з одним певними відносинами. Це можуть бути один-до-одного, один-до-багатьох або багато-до-багатьох.

Більшість ORM-інструментів, таких як Entity Framework Core підтримують 2 основні типи генерації бази даних та моделей:

- Code first
- Database first

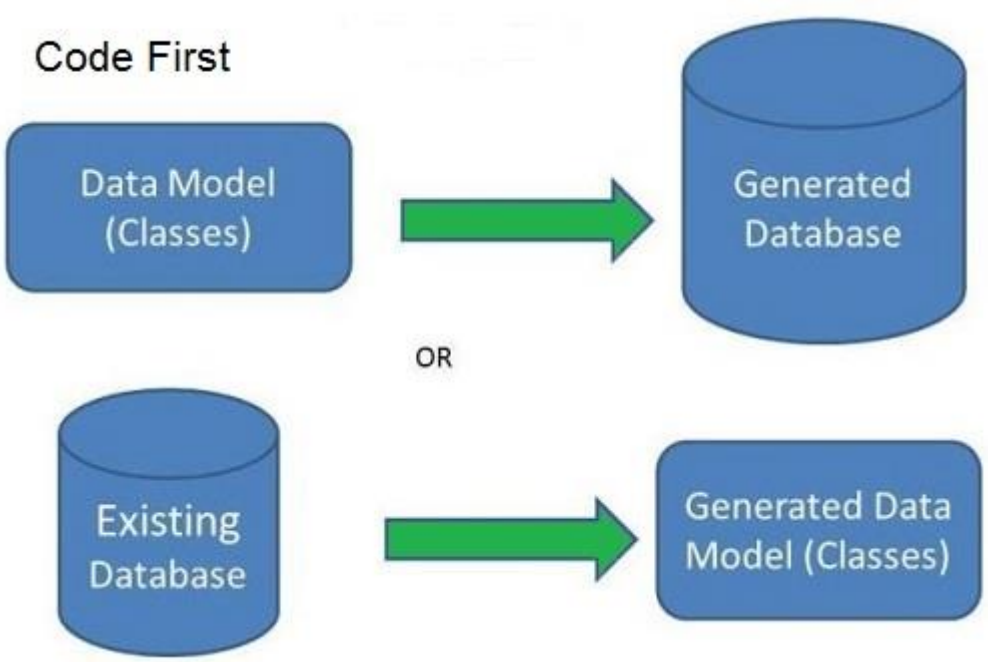


Рисунок 4.1

Було вирішено використовувати принцип code first, так як характеристики об’єктів зрозумілі та описати модель не складало проблеми.

Для початку було відокремлені 3 головні таблиці в системі обліку надзвичайних ситуацій :

1. Надзвичайна ситуація

Ця таблиця відповідала за клас НС: техногенний, природній, соціальний, воєнний і т.д.

2. Подія

Ця таблиця містить в собі головні показники кожної події, що сталась. Є така інформація як назва події, опис події, кількість жертв, кількість постраждалих, збитки, кошти на відновлення, дата події та зображення з місця подій.

3. Місце події

Ця таблиця зберігає координати події на карті та словесний опис місця.

Таблиці між собою відносяться так:

- Надзвичайна ситуація – подія один-до-багатьох
- Подія – місце події один-до-одного

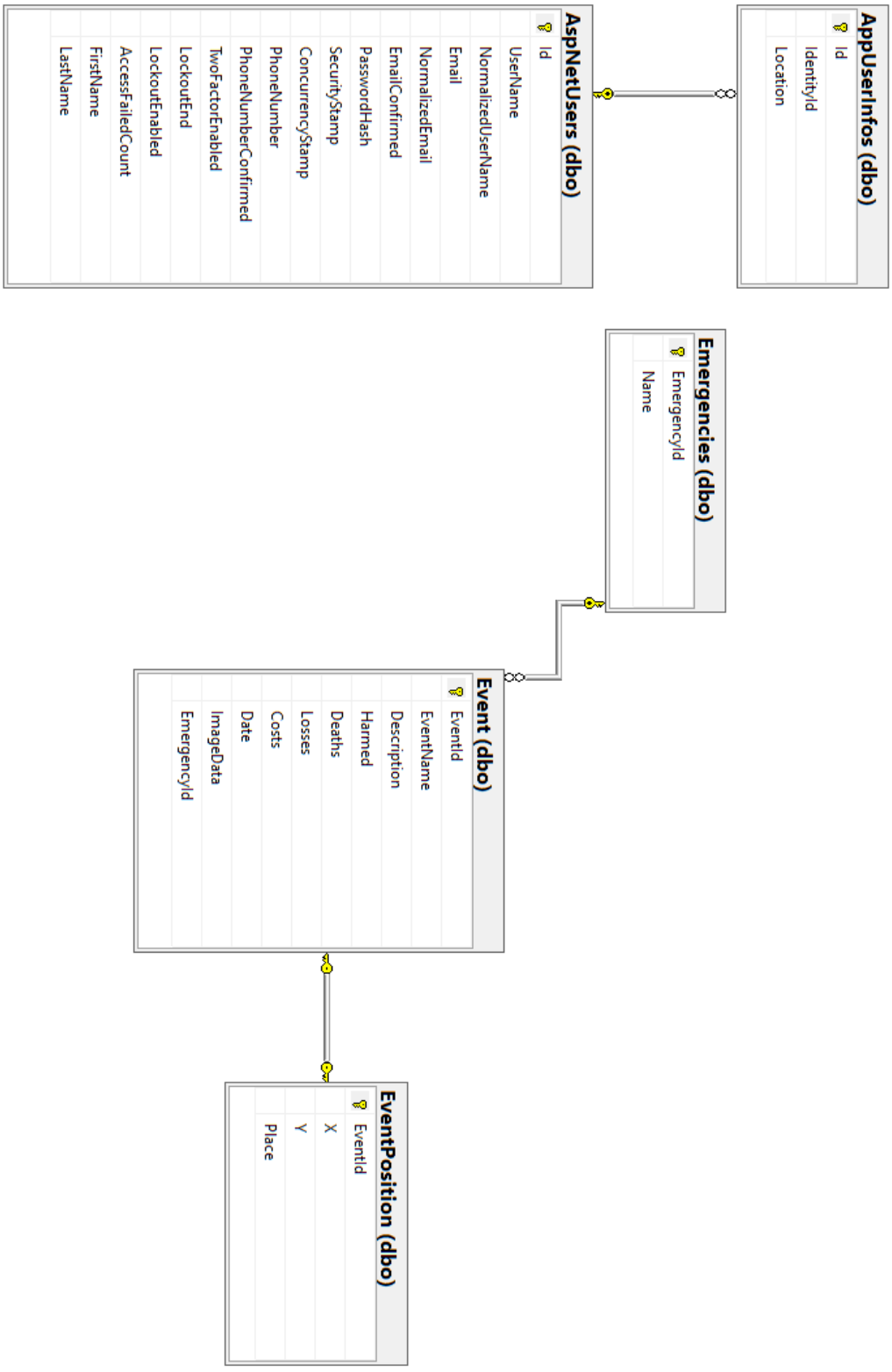


Схема 4.1 схема БД

Також в базі даних містяться 2 таблиці, які створені системою .NET Core Identity. В них зберігається інформація про користувачів.

4.2 Реалізація серверної частини

Серверна частина системи обліку надзвичайний ситуацій реалізована на .NET Core на мові C#.

Розробка велась з дотриманням рекомендацій «чистого коду» та з дотриманням принципів SOLID. Дотримання всіх цих принципів робить програмний продукт легким в масштабуванні та подальшій підтримці.

В процесі написання проводили юніт тестування, для забезпечення правильного функціонування кожної окремої частини.

Більшість серверної частини складають контролери для обробки запитів з клієнтської частини та моделі бази даних.

Реалізовано такі контролери:

- AuthController для роботи з авторизацією адміністратора, отримання токена, з яким надалі мають виконуватися запити від авторизованого користувача
- EmergencyController для отримання інформації про конкретній тип надзвичайної ситуації, та всі її події
- EventController для отримання надзвичайних подій та таких операцій над ними як додавання та редагування

Також реалізовані моделі для роботи з базою даних. Моделі відповідають таблицям:

- Emergency
- Event
- EventPosition

4.3 Реалізація клієнтської частини

Реалізація клієнтської частини представляє собою Angular додаток. Це тип веб додатку, на основі TypeScript. При створенні було використано версію Angular 8.0. Це версія від Травня 2019 року. Та TypeScript 3.4.

При розробці було створено декілька модулів, головні з них відповідають за сторінки:

- Інформації на карті
- Інформації в таблицях
- Додання нової події
- Авторизації

Для відображення інформації на карті застосовано карту Google. Google карти це безкоштовний картографічний сервіс. У ньому зібрані карта та супутникові знімки планети Земля. Для розробників сторонніх сервісів Google Maps дають можливість використовувати сервіс за основу. Google створили API для інтеграції сервісу та детальний опис підключення до своєї системи.

Для використання карти у системі обліку надзвичайних ситуацій ми пройшли певні кроки:

1. Створити запит на отримання API ключа від компанії Google;
2. Встановити Angular Google Maps, це набір директив для успішної інтеграції сервісу Google Maps з Angular додатком;
3. Переглянувши документацію на офіційному сайті Google, створити власну потрібну конфігурацію

4.4 Налагодження зв’язку між сервером та клієнтом

Для спілкування клієнт сервер використовується HTTP.

Більшість прикладних додатків для зв’язку з сервером використовують протокол HTTP, щоб приймати або віддавати інформацію з серверу. Angular забезпечує спрощений клієнтський HTTP API для кутових додатків, клас обслуговування HttpClient у @ angular / common / http.

Головні особливості:

- Можливість отримувати об’єкти потрібного типу (як примітивні, так і складні).
- Спрощене поводження з помилками.
- Особливості тестування.
- Перехоплення запиту та відповіді.

Клієнт має всі основні методи запитів. Найпоширеніший метод get(), для отримання інформації з серверу. HttpClient підтримує й інші методи HTTP, такі як PUT, POST та DELETE, які можна використовувати для зміни віддалених даних [8].

4.5 Робота з авторизацією

Авторизація в системі потрібна для того, щоб була можливість додати нову подію. Можливість додати подію не може бути відкритою для всіх, адже тоді буде багато недостовірної інформації та сервіс не матиме сенсу.

Отже, для авторизації було використано JSON Web Tokens.

Після того, як користувач ввійде в систему використовуючи свої дані (такі як адреса пошти та пароль), йому у відповідь від серверу прийде JSON Web Token.

Надалі, кожного разу коли користувач захоче отримати доступ до сторінки додання події агент користувача повинен надіслати JSON Web Token у заголовку авторизації. Якщо токен буде відсутній в заголовках, то користувач не можу отримати дані сторінки.

Спілкування відбувається як на рисунку нижче:

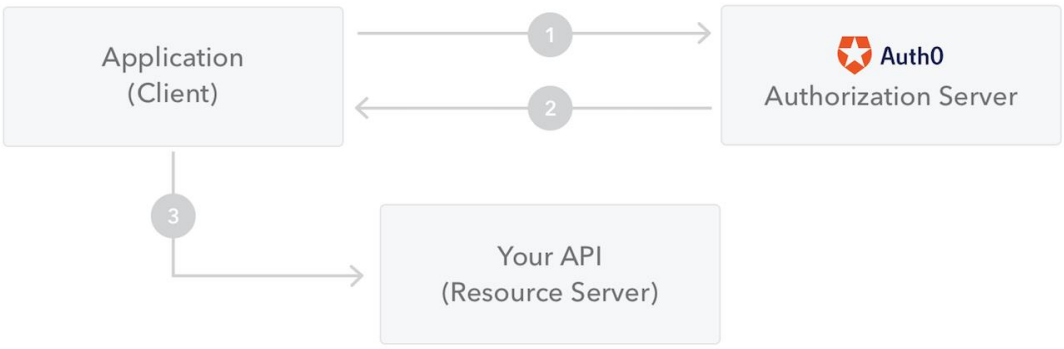


Схема 4.2 Робота з JSON Web Token.

1. Додаток або клієнт вимагає авторизації на сервері авторизації. Це виконується через один з різних потоків авторизації. Наприклад, типовий веб-додаток, сумісний з OpenID Connect, пройде через кінцеву точку, використовуючи потік коду авторизації.

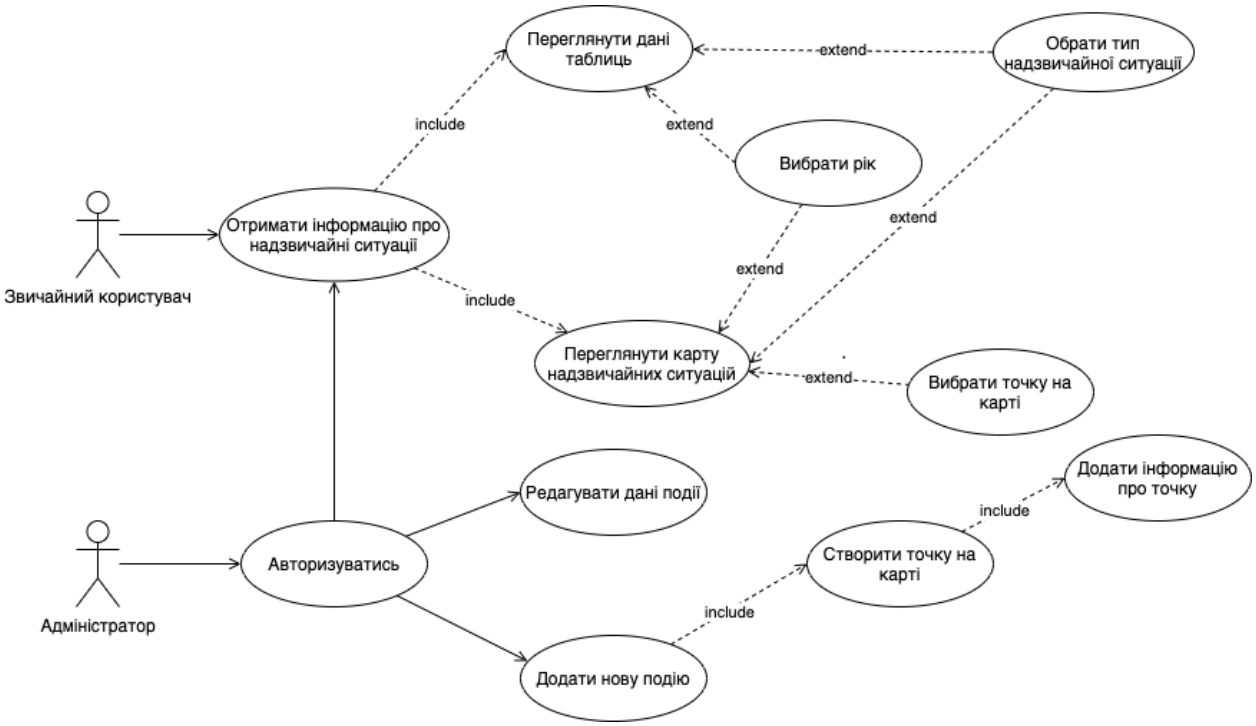
2. Коли авторизація надана, сервер авторизації повертає маркер доступу до програми.

3. Додаток використовує маркер доступу для доступу до захищеного ресурсу (наприклад, API) [9].

5. РОБОТА КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Для того щоб користуватися створеною системою обліку надзвичайних ситуацій природного характеру необхідно мати доступ до мережі Internet та браузер (підтримуються всі популярні браузери).

Діаграма президентів виглядає так:



Діаграма 1 Діаграма прецедентів

Першою сторінкою, що бачить користувач є сторінка з картою, на якій зображені всі події за всі роки :

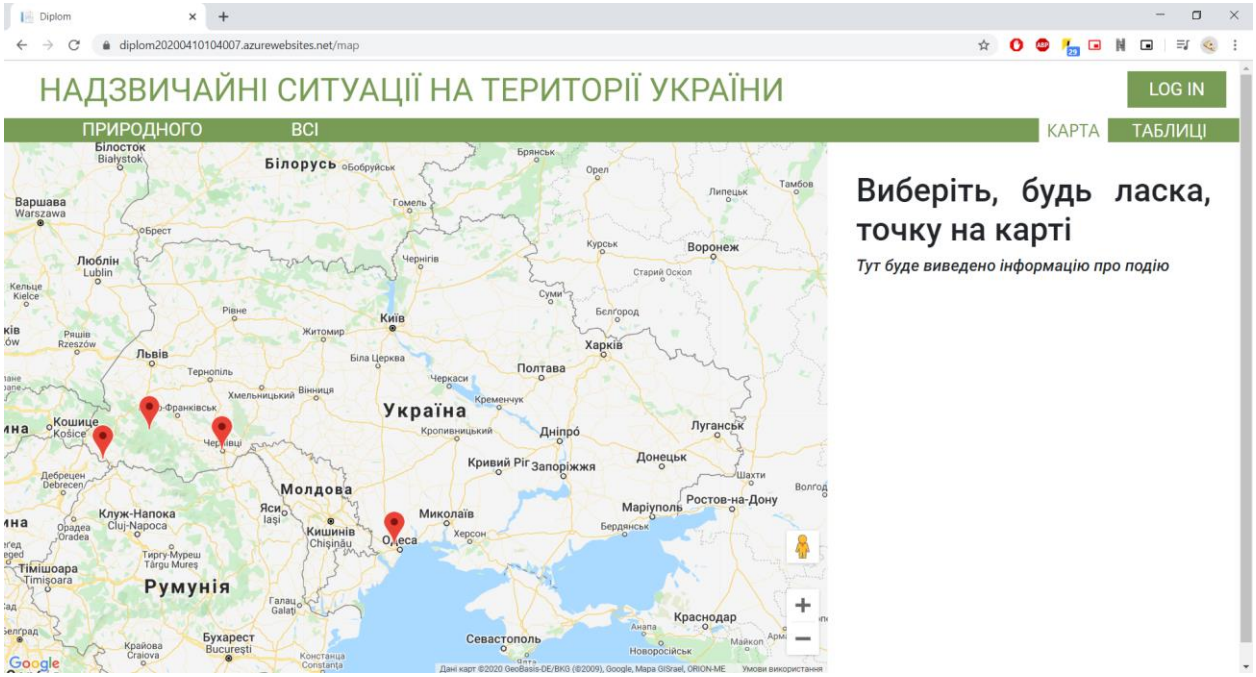


Рисунок 5.1 Початкова сторінка програми

Є можливість збільшити та зменшити масштаб карти:

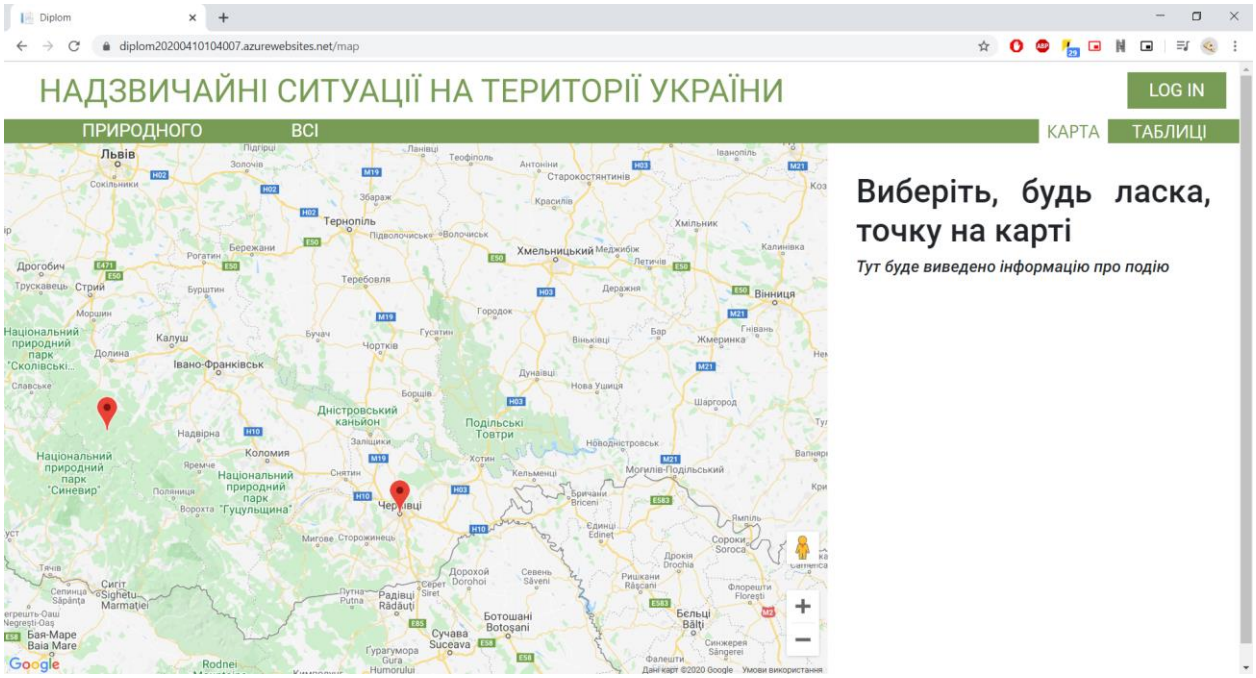


Рисунок 5.2а Збільшення карти

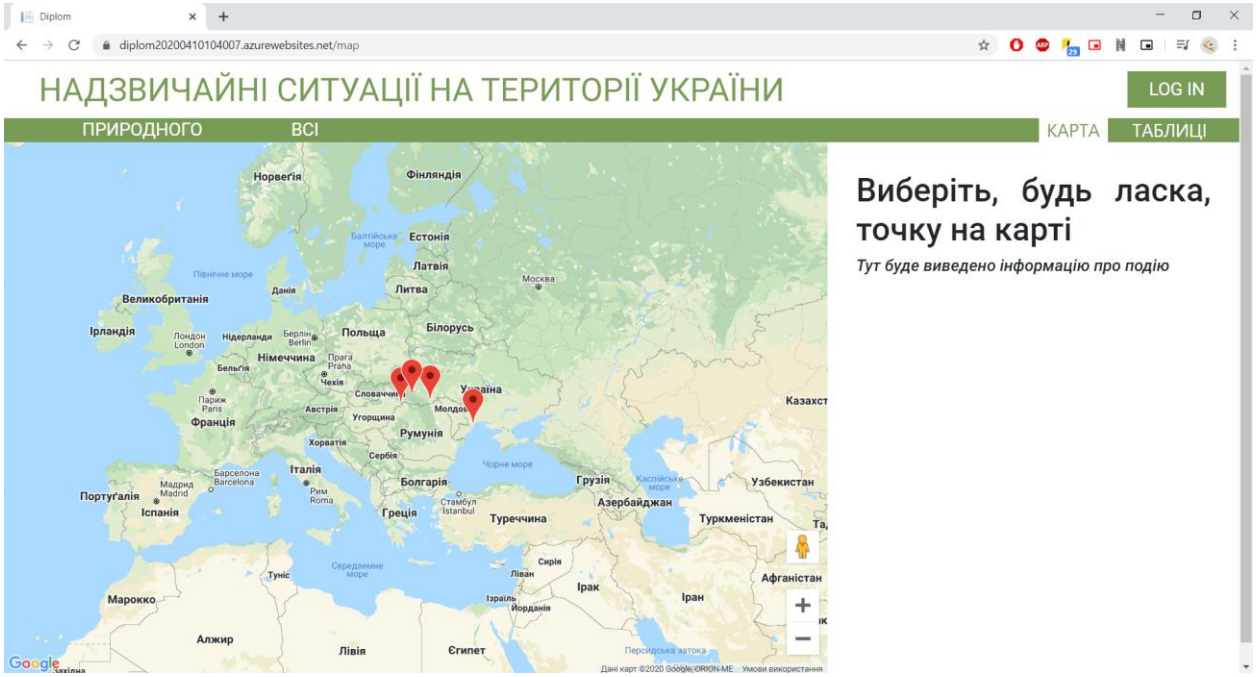


Рисунок 5.2б Зменшення карти

В Лівому верхньому кутку можна вибрати рік, який цікавить:

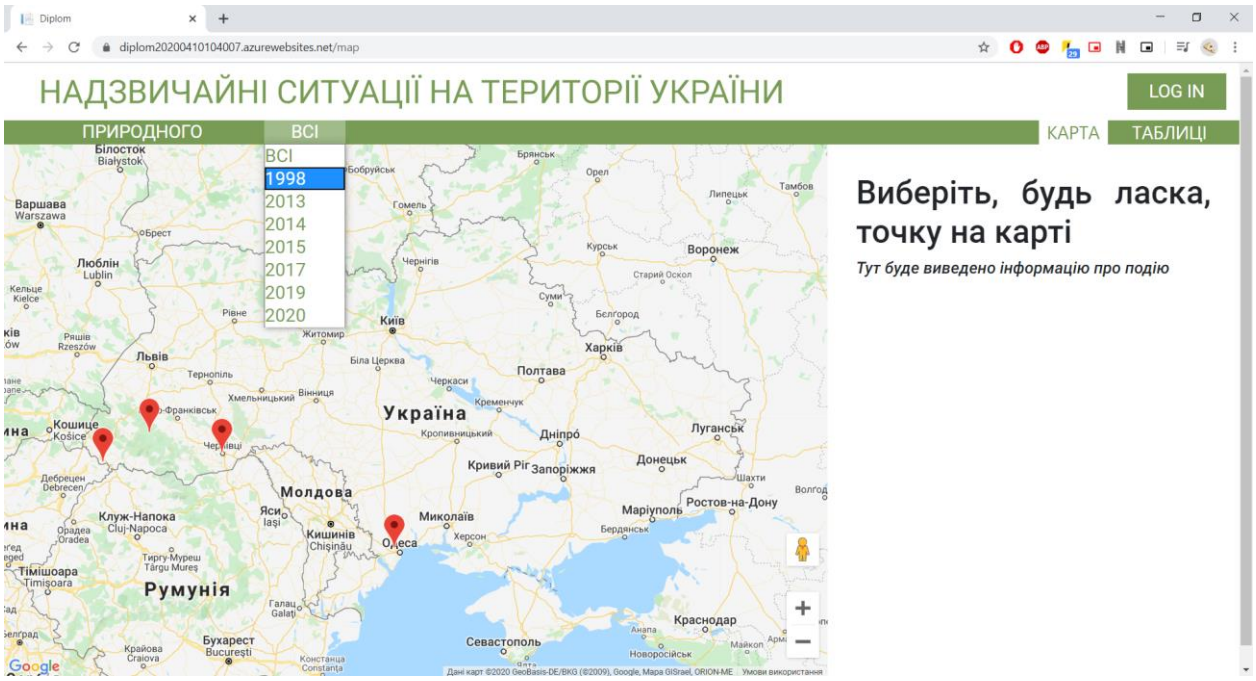


Рисунок 5.3 Вибір року події

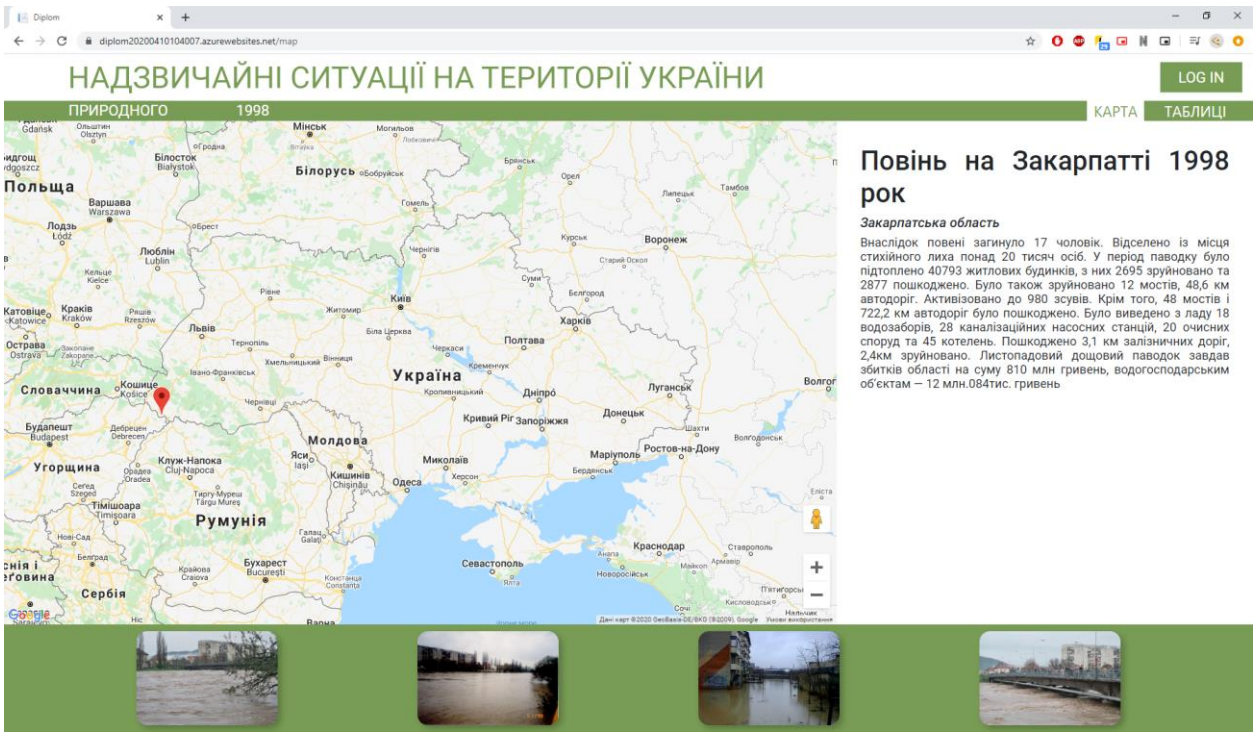


Рисунок 5.4 Результат пошуку за роком, в 1998 тільки одна НС занесена до бази даних

Коли знайшли подію яка цікавить можна натиснути на макер, що позначає цю подію на карті. Після цього з правого боку від карти з'явиться короткий опис подій:

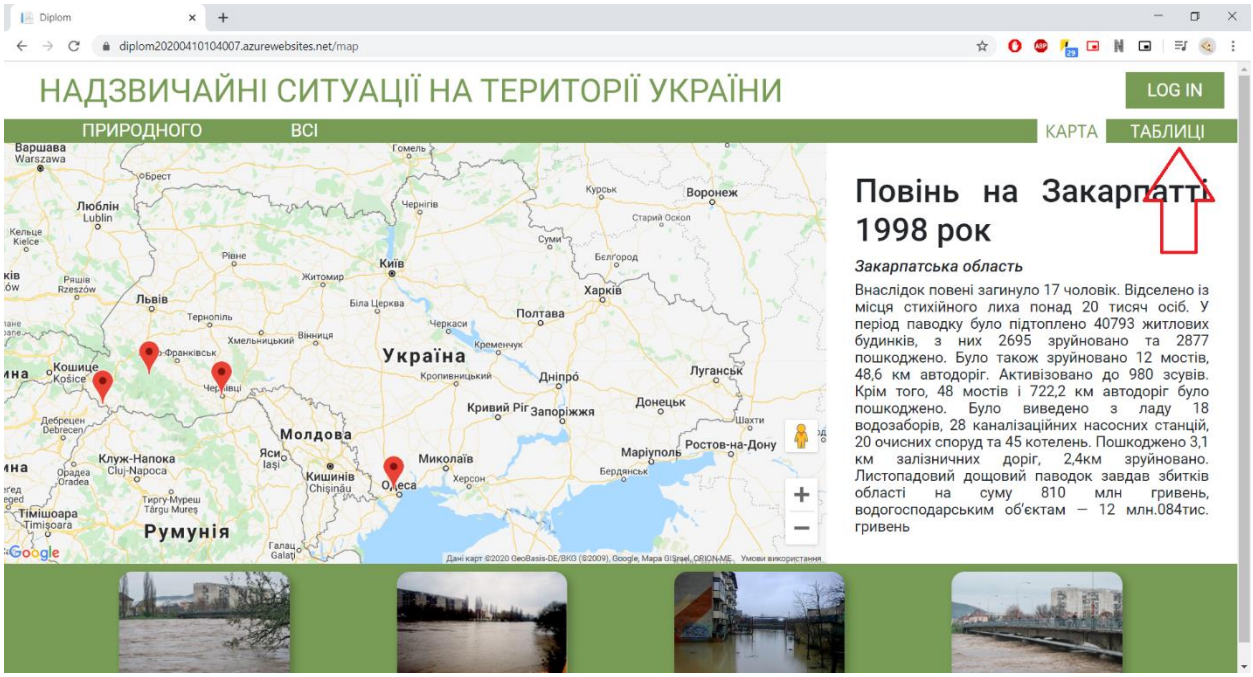


Рисунок 5.5 Кнопка перемикання режимів перегляду

Цю ж інформацію але у вигляді таблиці можна переглянути натиснувши на напис «Таблиці» у правому верхньому куртку:

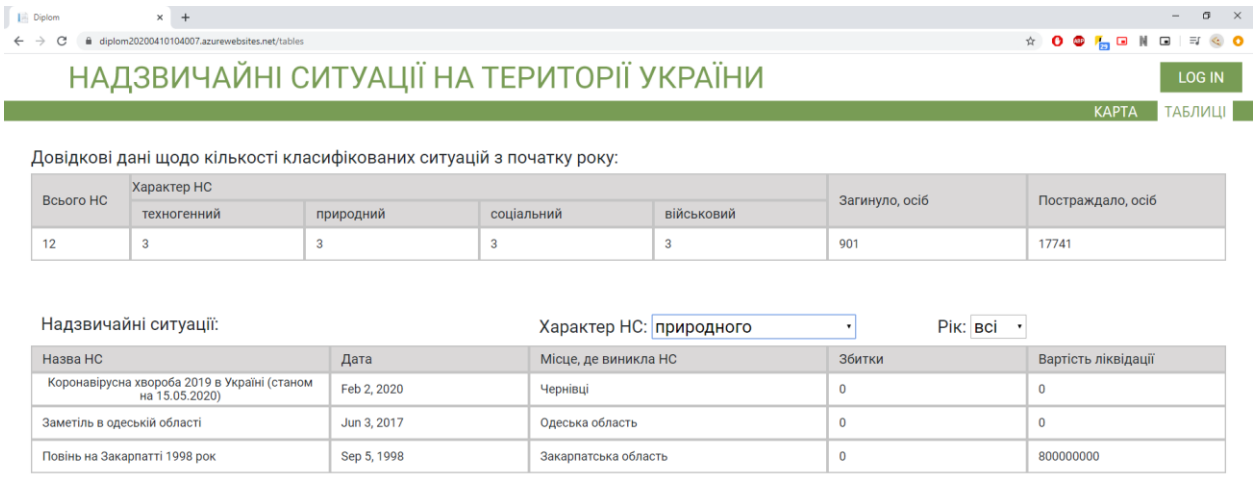


Рисунок 5.6 Перегляд в табличному представленні

Над другою таблицею є такі ж випадаючи списки для фільтрації, аналогічні як на карті:

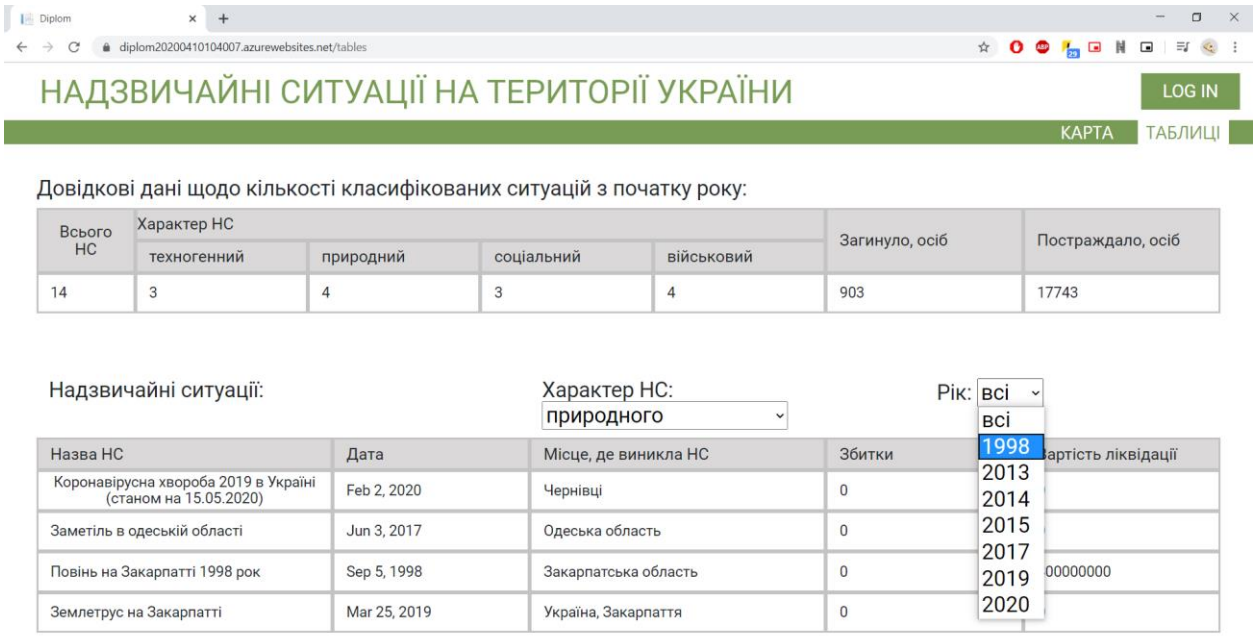


Рисунок 5.7 Вибір року події в табличному представленні

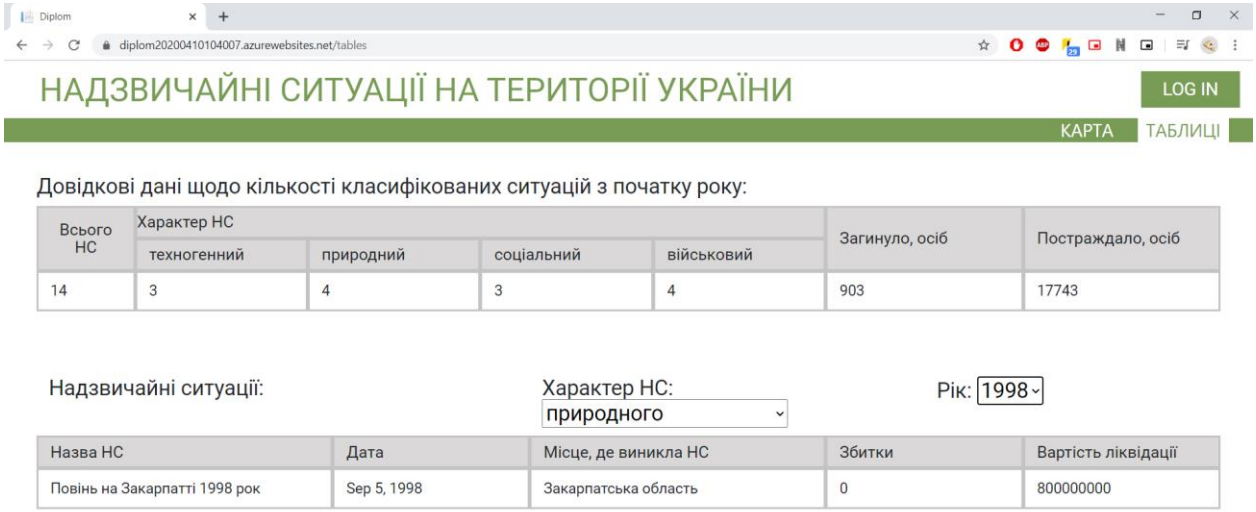


Рисунок 5.8 Результат пошуку за роком в табличному представленні , в 1998 тільки одна НС занесена до бази даних

З рисунків вище зрозуміло, що на першій таблиці на сторінці наведені певні статистичні дані. Підписані колонки відповідно.

Якщо адміністратор хоче додати нову подію він повинен натиснути на кнопку в правому верхньому кутку та увійти в систему:

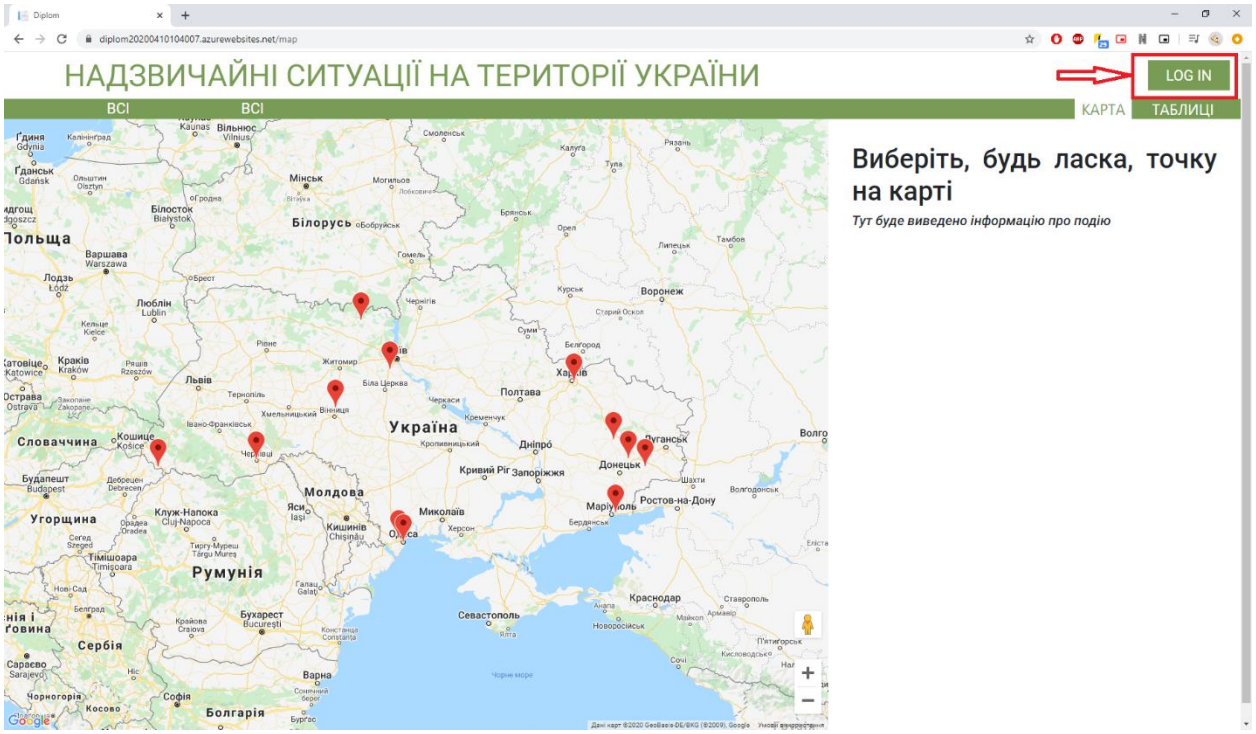


Рисунок 5.9 Перехід на сторінку авторизації

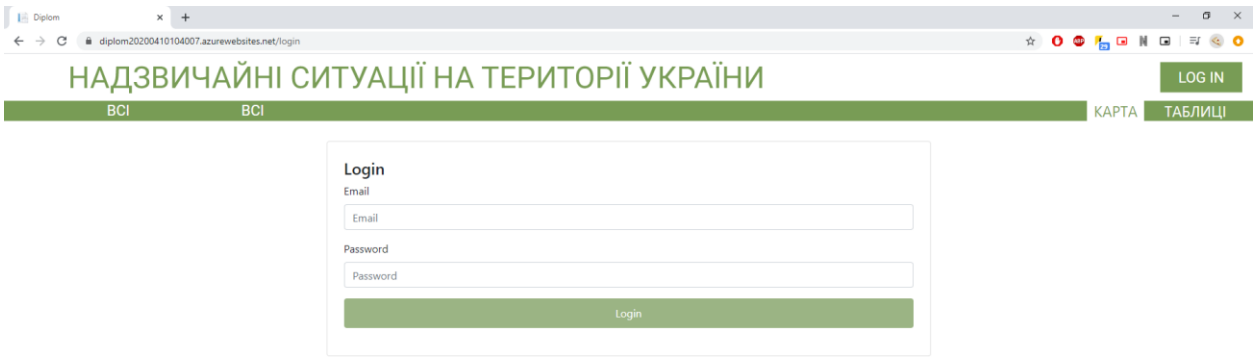


Рисунок 5.10 Сторінка авторизації адміністратора

При вводі некоректних даних користувач отримає повідомлення. Основні валідації:

- Ввід поштового адресу, тобто повинен бути знак @
- Пароль не менше 6 символів

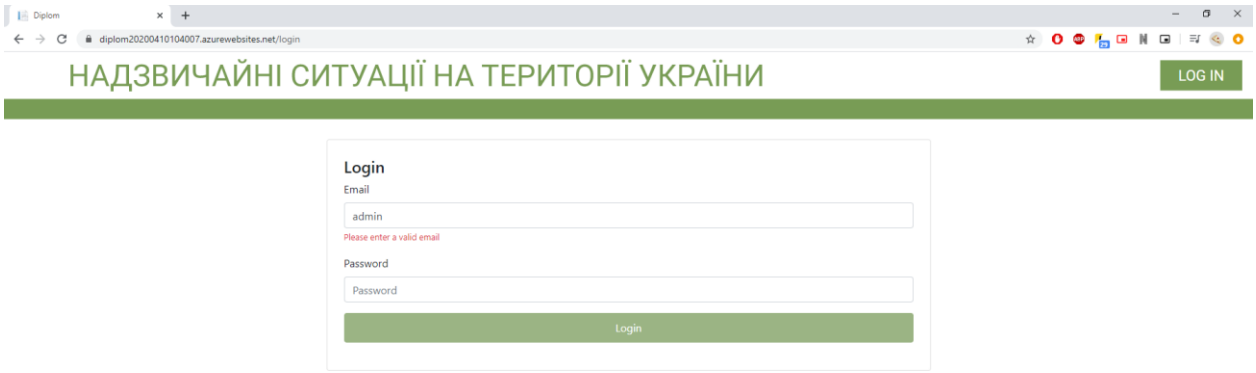


Рисунок 5.11 Перевірка валідності поля електронної адреси

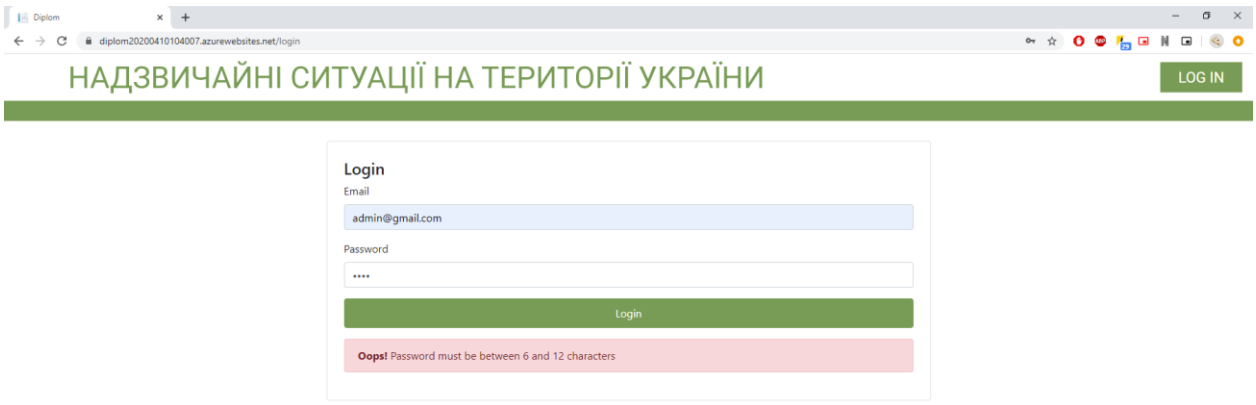


Рисунок 5.12 Перевірка валідності поля пароля

При вводі некоректних даних очікується помилка

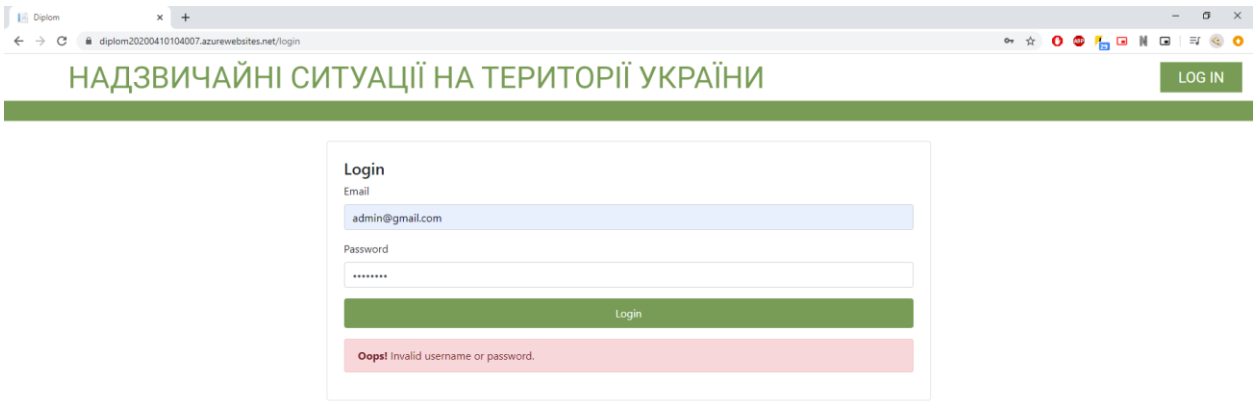


Рисунок 5.13 Перевірка користувача в базі даних

Після вводу коректних даних він може додати подію

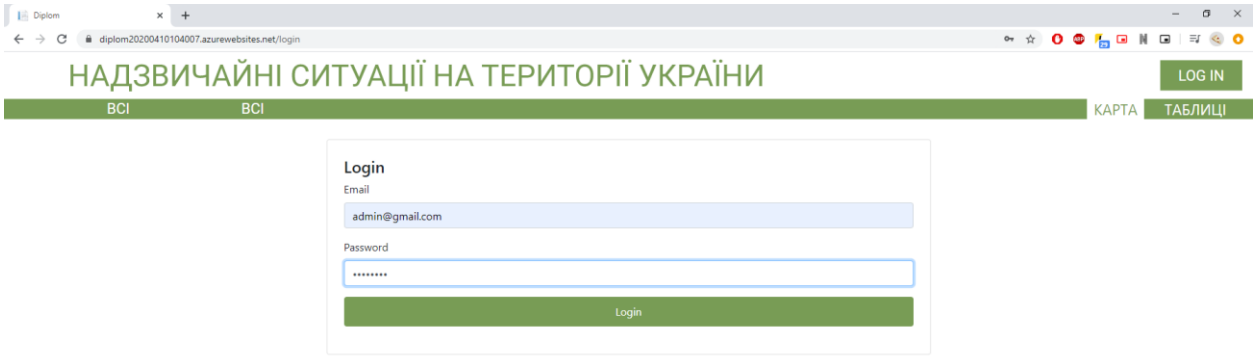


Рисунок 5.14 Успішна авторизація

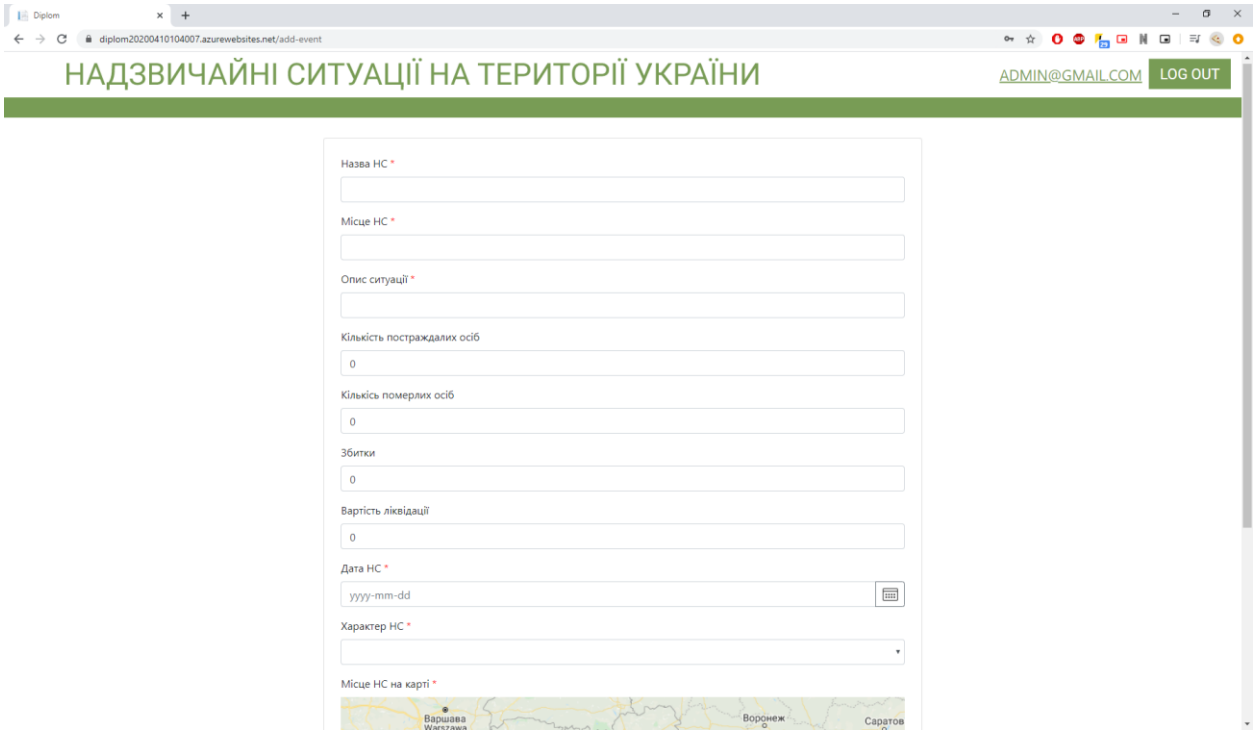


Рисунок 5.15 Сторінка додання нової події

Для додання нової подію потрібно заповнити форму. Поля без яких форму неможливо зберегти позначені червоною зірочкою.

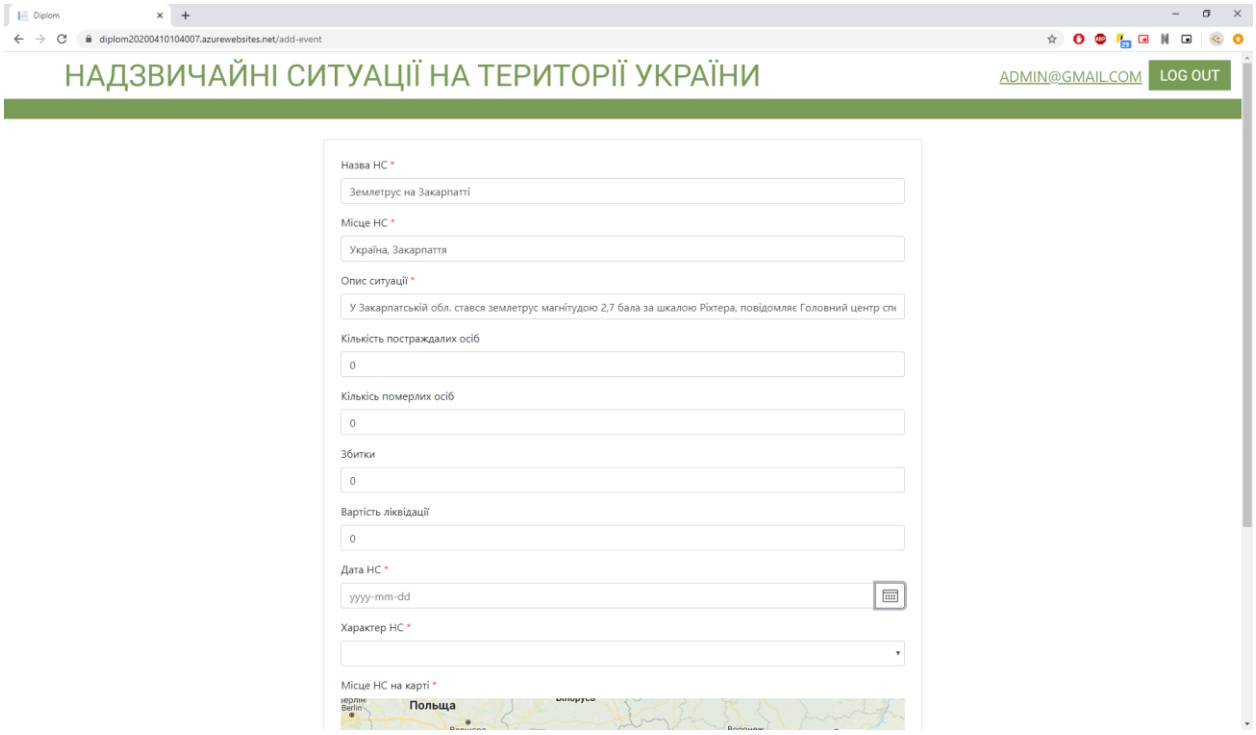


Рисунок 5.16 Приклад заповнення полів нової події

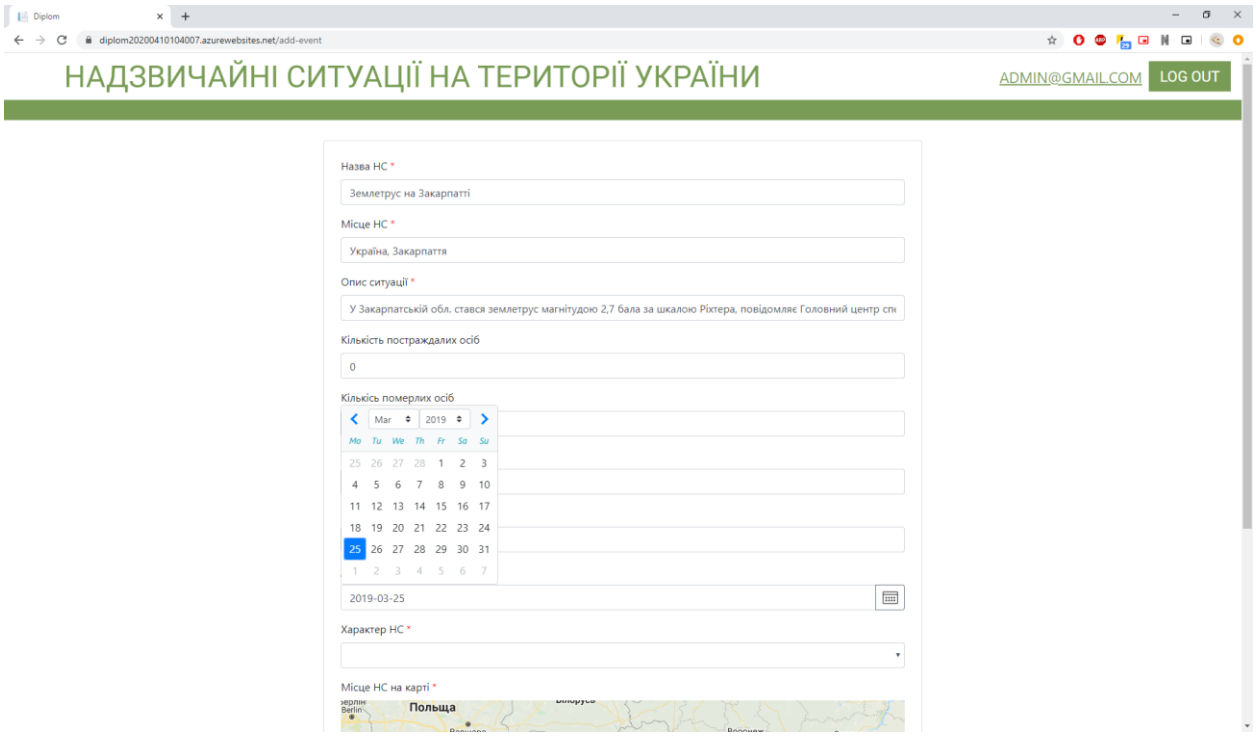


Рисунок 5.17 Приклад заповнення дати нової події

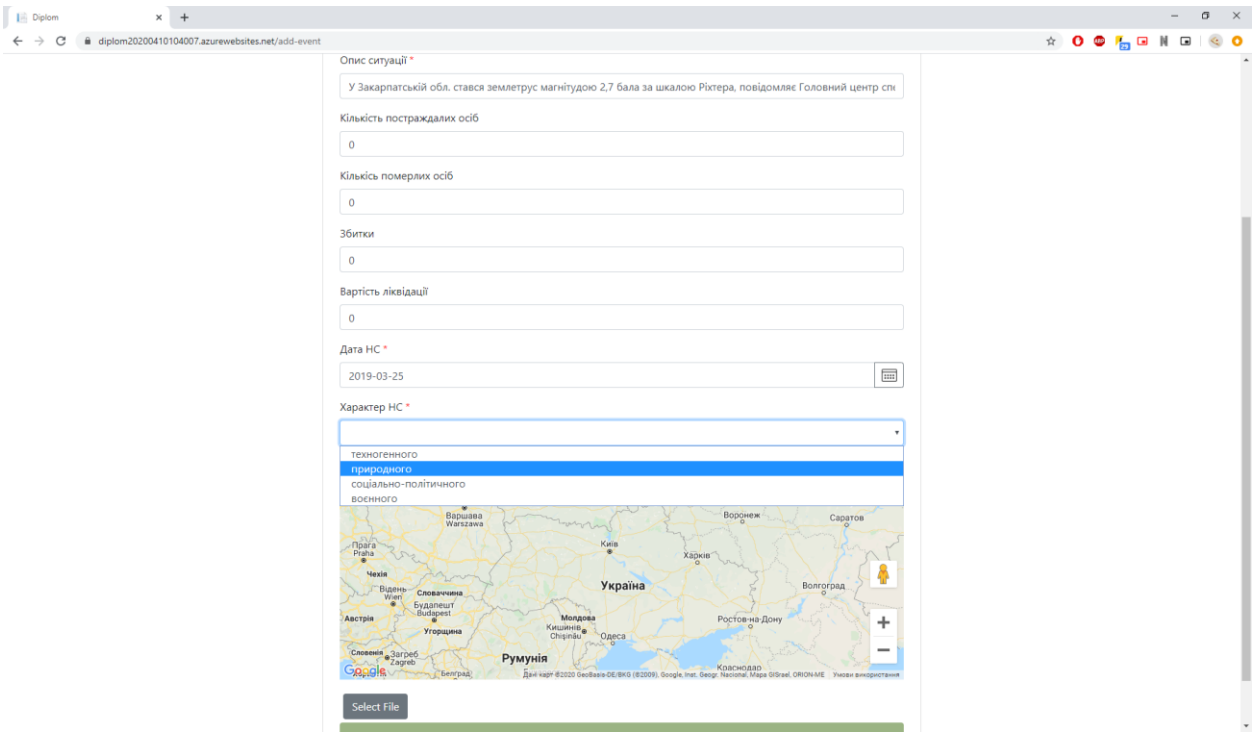


Рисунок 5.18 Приклад заповнення характеру нової події

Для того, щоб поставити точку на карті, потрібно натиснути на те місце в якому сталась подія:

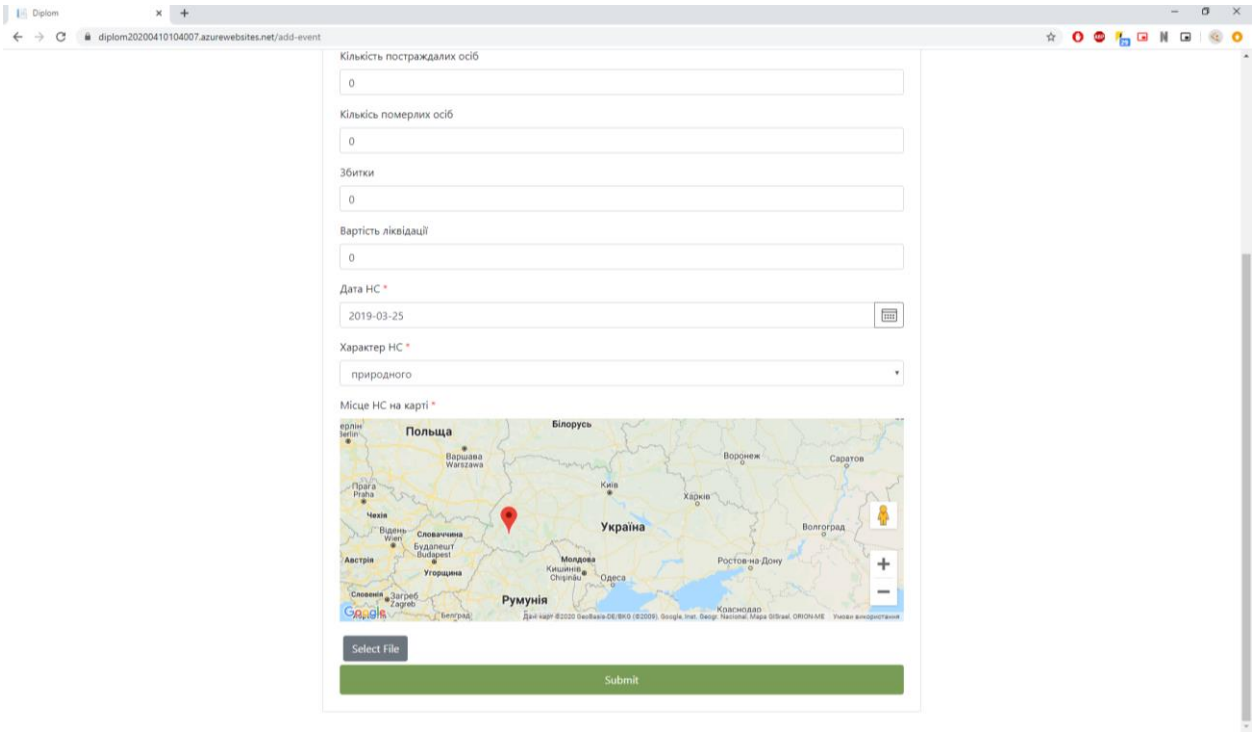


Рисунок 5.19 Приклад заповнення локації нової події

Також є можливість до кожної події додати фото очевидців:

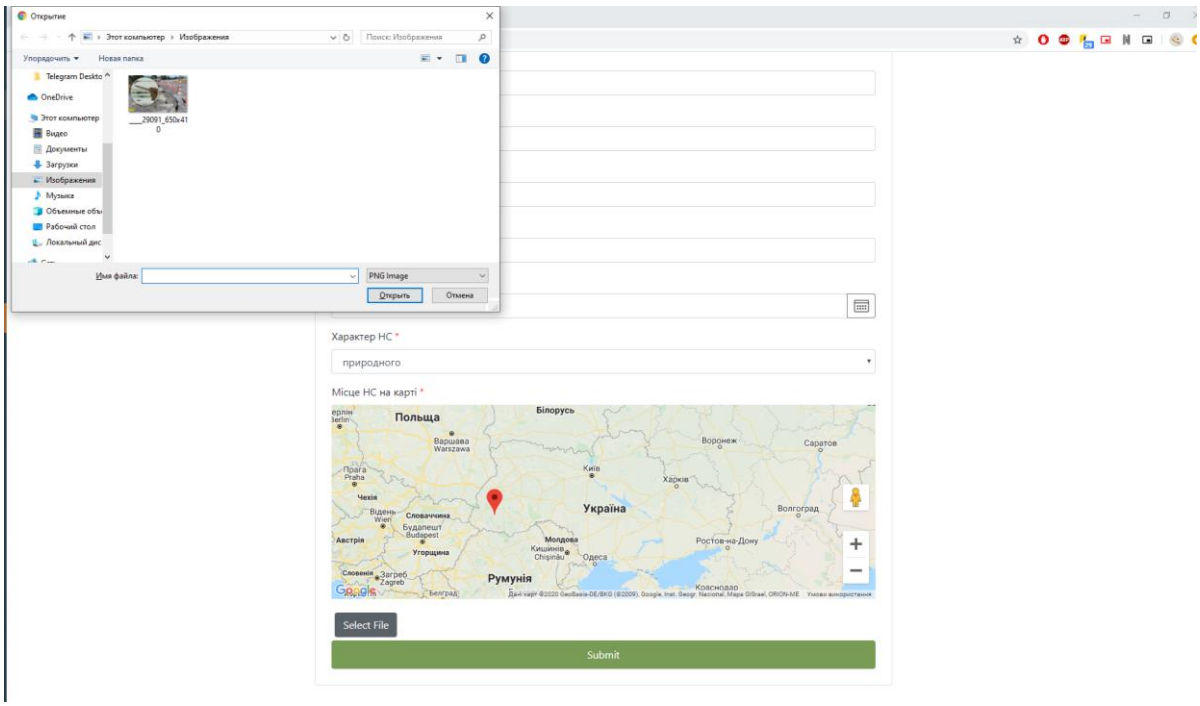


Рисунок 5.20 Приклад додавання нових фото події

Після заповнення всіх необхідних полів форми, з’являється можливість натиснути на кнопку «Submit». До заповнення всіх полів з зірочкою, додати подію неможливо. Після успішного додання події, сторінка переходить на карту, де вже можна побачити щойно додану подію.

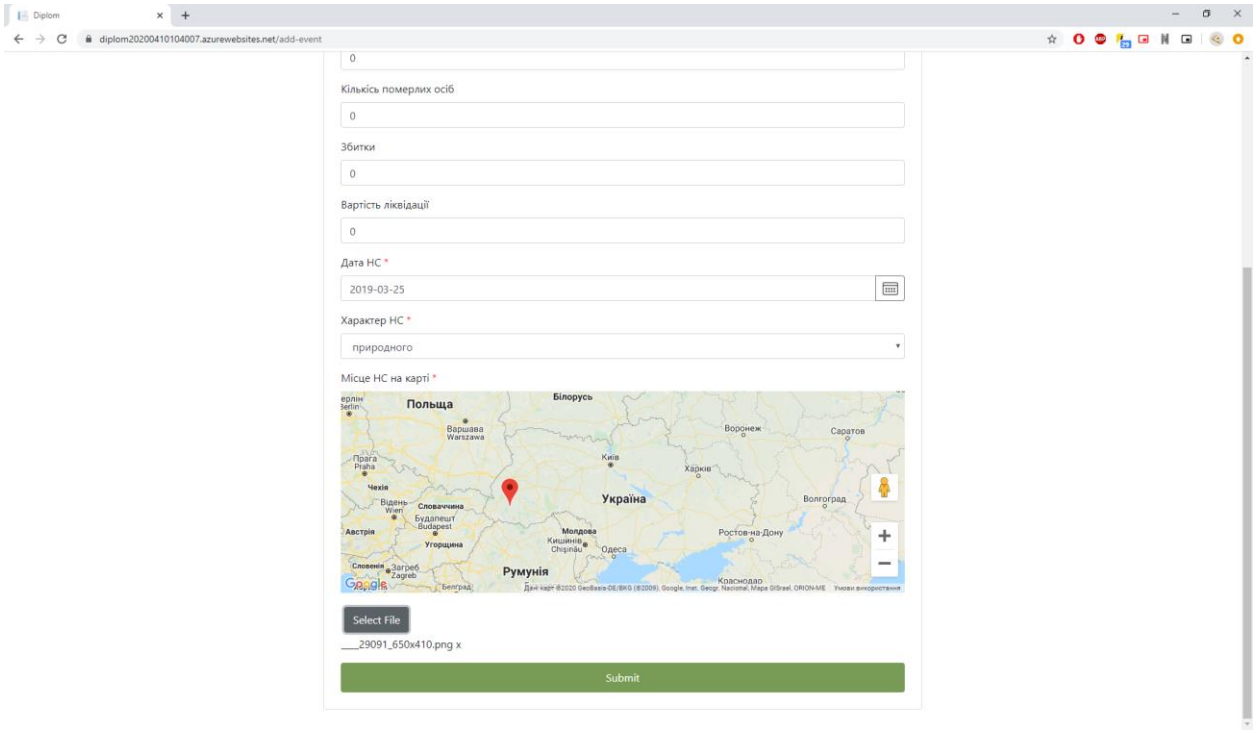


Рисунок 5.21 Збереження події

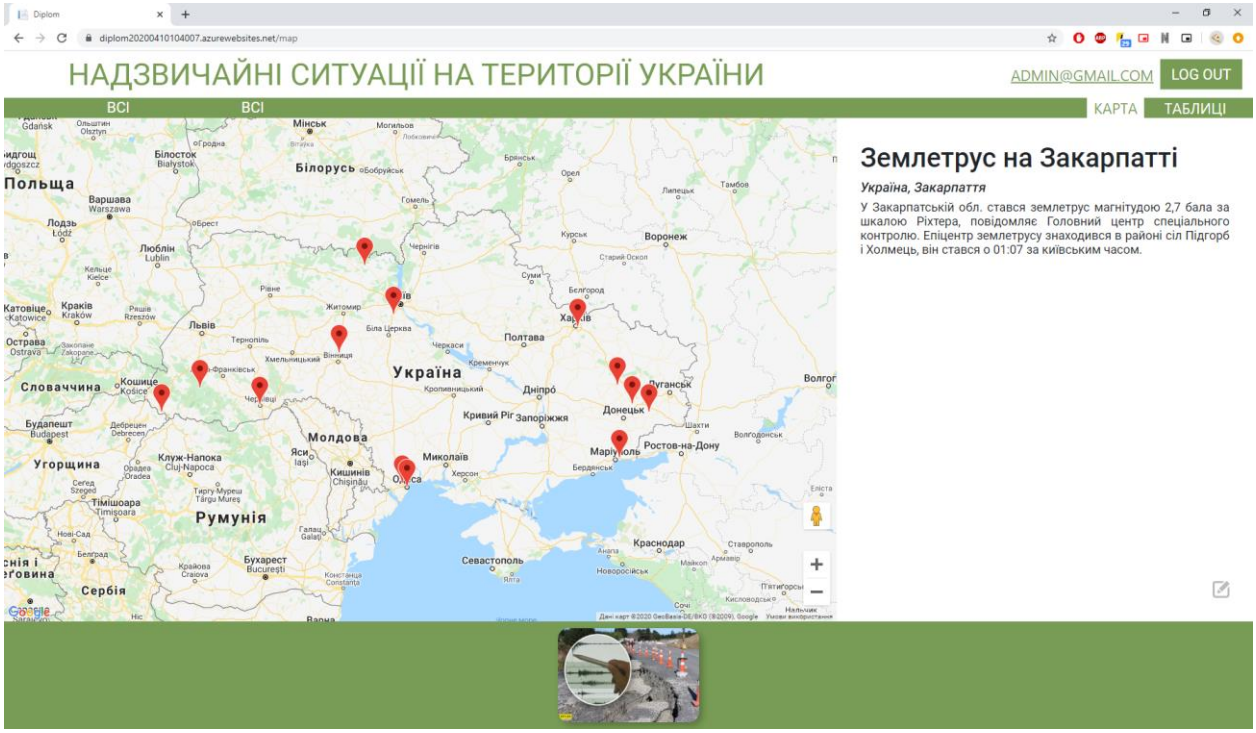


Рисунок 5.22 Відображення новоствореної події на карті

Якщо після додання події, було помічено помилку в даних, або виявились нові факти, то адміністратор має можливість відредагувати подію. Для цього необхідно натиснути на кнопку редагування в нижньому правому кутку на сторінці з картою:

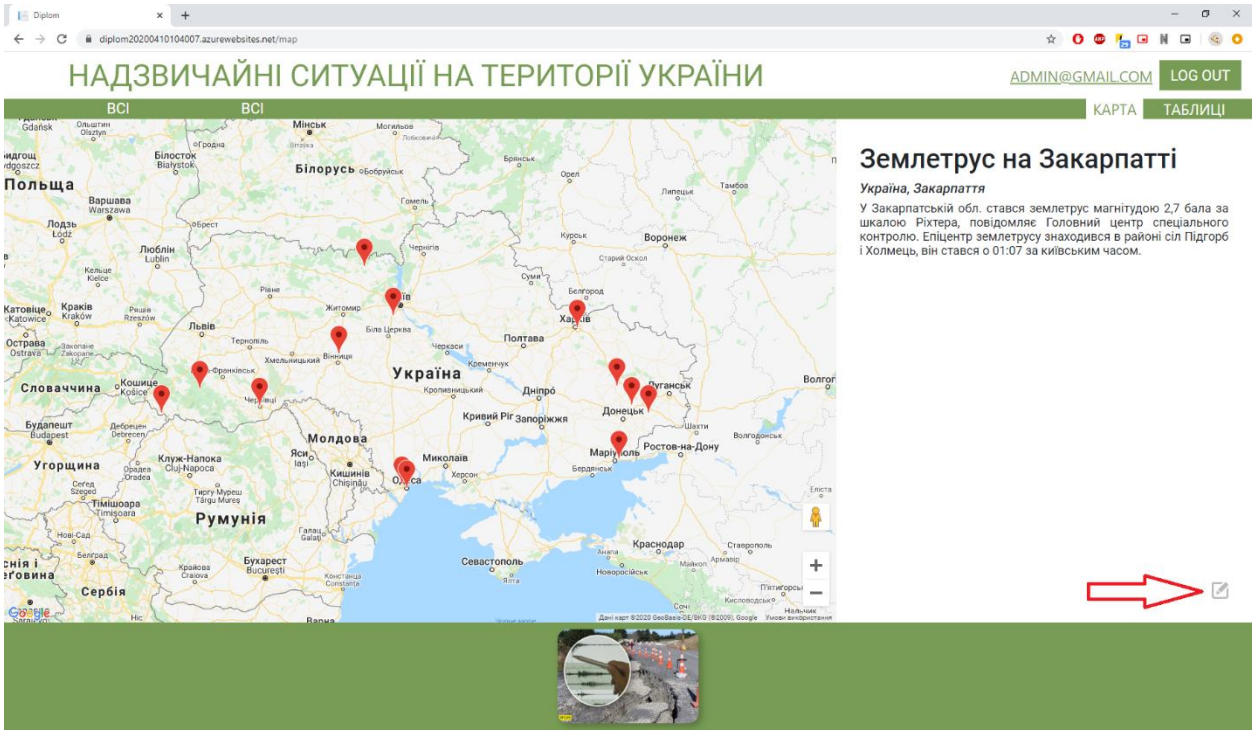
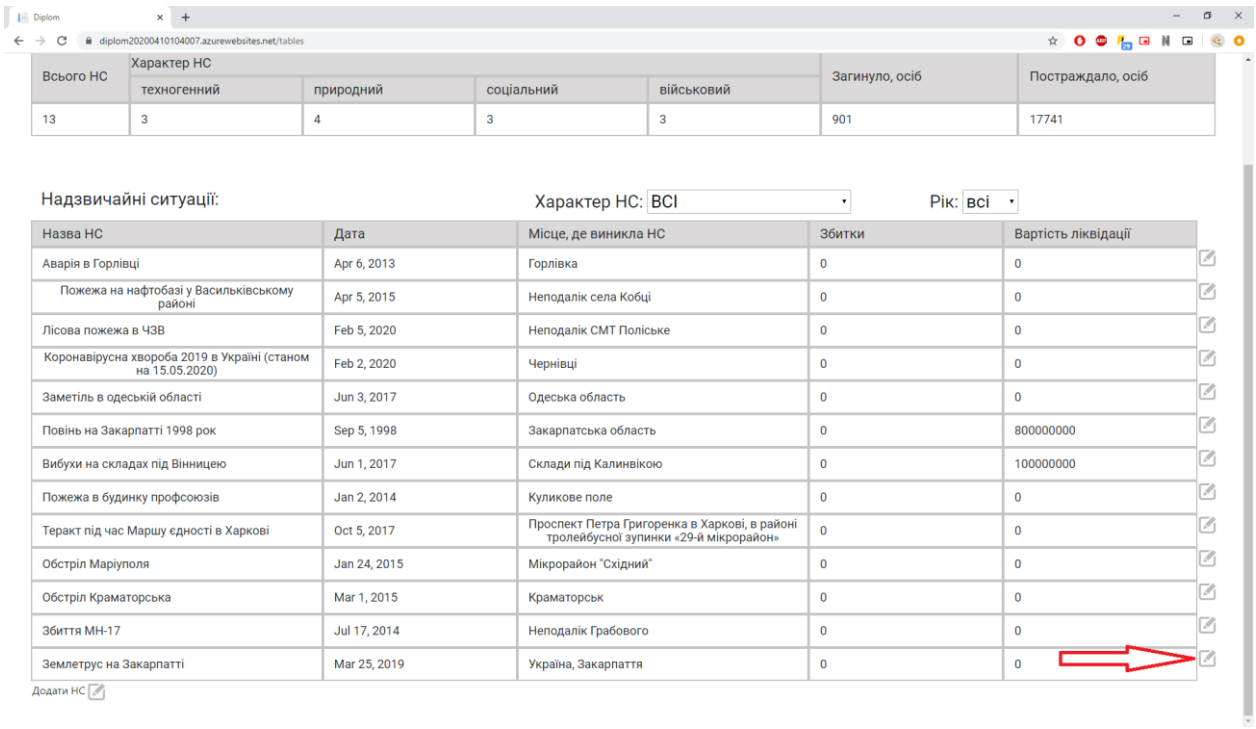


Рисунок 5.23 Редагування існуючої події

Або на таку ж кнопку біла події в табличному вигляді



Всього НС	Характер НС				Загинуло, осіб	Постраждало, осіб
	техногенний	природний	соціальний	військовий		
13	3	4	3	3	901	17741

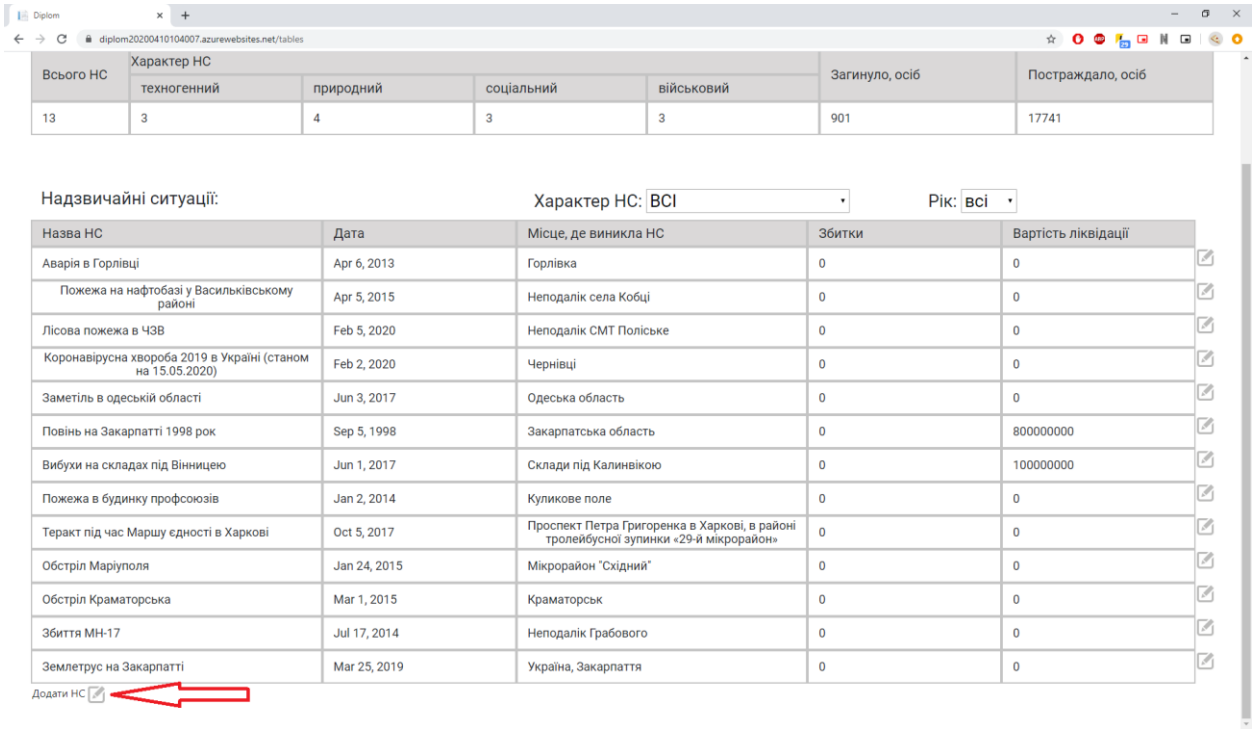
Надзвичайні ситуації: Характер НС: ВСІ Рік: всі

Назва НС	Дата	Місце, де виникла НС	Збитки	Вартість ліквідації
Аварія в Горлівці	Apr 6, 2013	Горлівка	0	0
Пожежа на нафтобазі у Васильківському районі	Apr 5, 2015	Неподалік села Кобці	0	0
Лісова пожежа в ЧЗВ	Feb 5, 2020	Неподалік СМТ Поліське	0	0
Коронавірусна хвороба 2019 в Україні (станом на 15.05.2020)	Feb 2, 2020	Чернівці	0	0
Заметіль в одеській області	Jun 3, 2017	Одеська область	0	0
Повінь на Закарпатті 1998 рок	Sep 5, 1998	Закарпатська область	0	800000000
Вибухи на складах під Вінницею	Jun 1, 2017	Склади під Калинівкою	0	100000000
Пожежа в будинку профсоюзів	Jan 2, 2014	Куликове поле	0	0
Теракт під час Маршу єдності в Харкові	Oct 5, 2017	Проспект Петра Григоренка в Харкові, в районі тролейбусної зупинки «29-й мікрорайон»	0	0
Обстріл Маріуполя	Jan 24, 2015	Мікрорайон "Східний"	0	0
Обстріл Краматорська	Mar 1, 2015	Краматорськ	0	0
Збиття МН-17	Jul 17, 2014	Неподалік Грабового	0	0
Землетрус на Закарпатті	Mar 25, 2019	Україна, Закарпаття	0	0

Додати НС

Рисунок 5.24 Редагування існуючої події в табличному вигляді

Додати нову подію також можна натиснувши кнопку «Додати НС» після таблиці з переліком подій. Таку можливість має тільки адміністратор.



Всього НС	Характер НС				Загинуло, осіб	Постраждало, осіб
	техногенний	природний	соціальний	військовий		
13	3	4	3	3	901	17741

Надзвичайні ситуації: Характер НС: ВСІ Рік: всі

Назва НС	Дата	Місце, де виникла НС	Збитки	Вартість ліквідації
Аварія в Горлівці	Apr 6, 2013	Горлівка	0	0
Пожежа на нафтобазі у Васильківському районі	Apr 5, 2015	Неподалік села Кобці	0	0
Лісова пожежа в ЧЗВ	Feb 5, 2020	Неподалік СМТ Поліське	0	0
Коронавірусна хвороба 2019 в Україні (станом на 15.05.2020)	Feb 2, 2020	Чернівці	0	0
Заметіль в одеській області	Jun 3, 2017	Одеська область	0	0
Повінь на Закарпатті 1998 рок	Sep 5, 1998	Закарпатська область	0	800000000
Вибухи на складах під Вінницею	Jun 1, 2017	Склади під Калинівкою	0	100000000
Пожежа в будинку профсоюзів	Jan 2, 2014	Куликове поле	0	0
Теракт під час Маршу єдності в Харкові	Oct 5, 2017	Проспект Петра Григоренка в Харкові, в районі тролейбусної зупинки «29-й мікрорайон»	0	0
Обстріл Маріуполя	Jan 24, 2015	Мікрорайон "Східний"	0	0
Обстріл Краматорська	Mar 1, 2015	Краматорськ	0	0
Збиття МН-17	Jul 17, 2014	Неподалік Грабового	0	0
Землетрус на Закарпатті	Mar 25, 2019	Україна, Закарпаття	0	0

Додати НС

Рисунок 5.25 Додавання нової події

Також для того, щоб показати, що розроблений програмний продукти має можливість проводити розрахунки збитків, завданих досліджуваною НС, було створено форму розрахунку збитків від лісової пожежі.

Для визначення збитків необхідно вказати такі параметри як:

- Площа лісової ділянки, що вилучається або знищується, у гектарах
- Область України
- Група лісу до та після НС
- Коефіцієнт продуктивності лісів (для зручності, таблиця коефіцієнтів задана під формою (Рисунок 5.27))
- Коефіцієнт зниження продуктивності угіддя

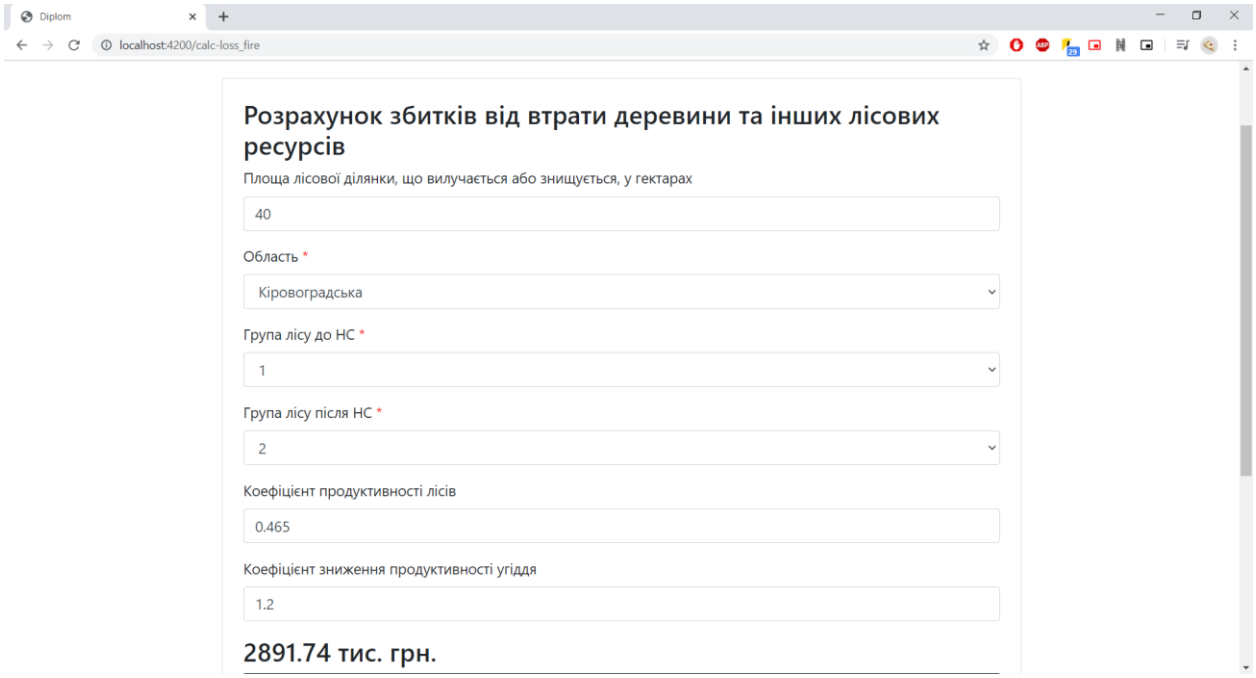


Рисунок 5.26 Приклад розрахунку збитків від втрати деревини та інших лісових ресурсів

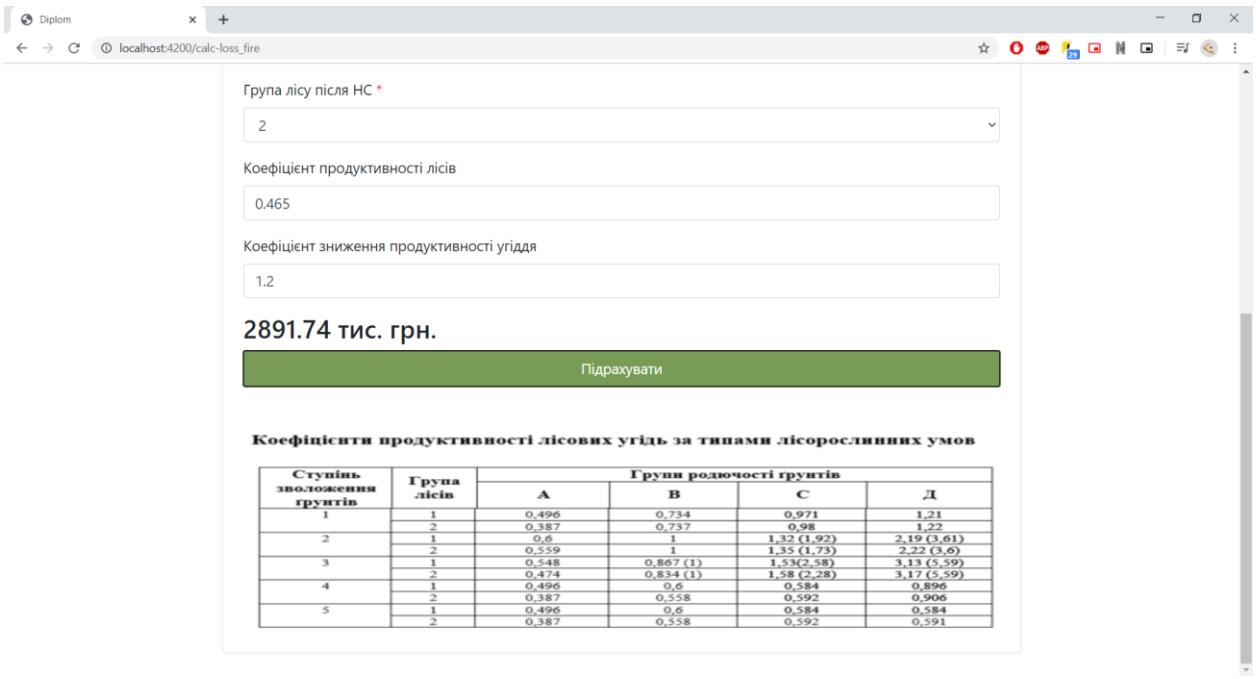


Рисунок 5.27 Таблиця коефіцієнтів продуктивності лісових угідь

ВИСНОВКИ

Виконуючи дану роботу було створено систему обліку надзвичайних ситуацій природного характеру. Систему створено за допомогою таких технологій як C#, CLR, .NET Core, EF Core, Azure Web Apps, SQL Server (SQL Azure), JWT authentication, Angular.

Систему розроблено з урахуванням поставлених на початку задач. В поточній реалізації є можливість створення нової та редагування існуючої події, перегляд всіх НС на карті та в табличному представленні. Окрім того, реалізований програмний продукт розташовано в глобальній мережі Інтернет, за допомогою Azure Web Apps, що робить його доступним для звичайного користувача без додаткових налаштувань.

Система є корисною як для звичайного громадянина, який хоче слідкувати за новинами, точніше за НС, в режимі карти, так і для уповноважених осіб, які можуть вплинути на ситуацію.

Проаналізувавши проблемні райони України, та НС які найчастіше трапляються можна краще слідкувати за станом прогнозовано небезпечних районів. Адже краще прикласти більше зусиль для того, щоб НС нанесло якомога менше шкоди, ніж пізніше боротися уже з колосальними наслідками.

Найближчі декілька століть навряд чи людство зможе приборкати природні сили, але ми вже зараз можемо створити умови, для того, щоб залишатися максимально в безпеці.

Розроблена система може стати основою для нових систем обліку. Програма може вдосконалюватися та зростати. Є можливість додати певний аналітичний функціонал та більше методів підрахунків збитків.

ДЖЕРЕЛА

1. Пожежа в Чорнобильській зоні [Електронний ресурс] – Режим доступу до ресурсу: https://tsn.ua/ru/ukrayina/pozhar-v-chernobyle-poslednie-podrobnosti-ob-epicentre-vozgoraniya-i-kuda-idet-ogon-1527291.html.
2. Надзвичайні ситуації природного характеру [Електронний ресурс] – Режим доступу до ресурсу: https://studfile.net/preview/5706637/.
3. Положення про Державну службу України з надзвичайних ситуацій [Електронний ресурс] – Режим доступу до ресурсу: https://www.dsns.gov.ua/ua/Polozhennya.html.
4. Документація по .NET Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-ru/dotnet/core/.
5. A tour of the C# language [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/.
6. SQL Server 2019 [Електронний ресурс] – Режим доступу до ресурсу: microsoft.com/en-us/sql-server/sql-server-2019.
7. Microsoft Azure [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Microsoft_Azure.
8. Angular FEATURES & BENEFITS [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/features.
9. Introduction to JSON Web Tokens [Електронний ресурс] – Режим доступу до ресурсу: https://jwt.io/introduction/.
10. Entity Framework Core [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/ef/core/.
11. Обзор Figma [Електронний ресурс] – Режим доступу до ресурсу: http://figmadesign.ru/1-0-0-obzor-figma.html.

Додаток 1

Система обліку надзвичайних ситуацій
природного характеру

Специфікація

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62

Аркушів 1

Київ 2020

Позначення	Найменування	Відмітки
Документація		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	Записка.docx	Пояснювальна записка
Компоненти		
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	Опис програмного модулю	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EmergenciesController.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EventsController.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	DiplomDatabaseContext.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	Emergency.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	Event.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EventPosition.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EventDTO.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	CredentialsViewModelValidator.cs	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	AddEventComponent.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	AddEventComponent.html	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	CalcLossFireComponent.html	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	CalcLossFireComponent.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	MainMapComponent.html	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	MainMapComponent.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	TablesPageComponent.html	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	TablesPageComponent.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EmergencyService.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	EventService.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	SelectorsService.ts	Основні компоненти

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	UserService.ts	Основні компоненти
УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62190_20Б	ConfigService.ts	Основні компоненти

Додаток 2

Система обліку надзвичайних ситуацій
природного характеру

Опис програмного модулю

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62

Аркушів 8

Київ 2020

АННОТАЦІЯ

Додаток містить опис програмної реалізації системи обліку надзвичайних ситуацій природного характеру. В додатку зображені файли, що відповідають:

- за зв’язок серверної та клієнтської частини;
- зв’язок серверної частини та бази даних;
- обробка та передача даних введених користувачем.

Також показані файли, що відповідають за головні сторінки користувацького інтерфейсу.

ЗМІСТ

ЗАГАЛЬНІ ВІДОМОСТІ

У цьому додатку відображено опис системи моніторингу надзвичайних ситуацій природного характеру.

Головні функціональні файли реалізованого програмного продукту описані в додатку 3.

При розробці програмного продукту серверну частину було розроблено за допомогою C# и .Net Core. Клієнтська частина розроблена у вигляді Angular додатку стилізованого Bootstrap. Для зберігання даних було обрано Microsoft Azure SQL Database. Реалізація авторизації втілена за допомогою jwt authentication.

ФУНКЦІОНАЛЬНЕ ПРИЗНАЧЕННЯ

Система обліку надзвичайних ситуацій природного характеру призначена як для пересічного користувача, що хоче просто дізнатися про надзвичайні ситуації природного характеру в Україні, так і для тих, хто може проаналізувати події та прийняти певні міри для того, щоб мінімізувати збитки, завдані НС.

Оскільки надзвичайні ситуації природного характеру не дуже часті, слідкувати за ними в щоденних новинах досить тяжко. Відображення подій в табличному представлені доцільне саме для цілей моніторингу, та зручна фільтрація за роками додає користувачу можливостей віднайти необхідну інформацію.

ВИКОРИСТОВУВАНІ ТЕХНІЧНІ ЗАСОБИ

Систему створено з використанням таких технологій як:

- C#,
- CLR,
- .NET Core,
- EF Core,
- Azure Web Apps,
- SQL Server (SQL Azure),
- JWT authentication,
- Angular,
- Bootstrap.

ВИКЛИК І ЗАВАНТАЖЕННЯ

Оскільки розроблений продукт є веб-сторінкою додаткових умов запуску немає. Завдяки використанню Azure Web Apps додаток було завантажено до всесвітньої мережі Інтернет і для того, щоб користуватися розробленим програмним продуктом необхідно лише ввести в пошуковий рядок браузера посилання https://diplom20200410104007.azurewebsites.net/map. Перейшовши на сторінку відкривається перша інформаційна сторінка з картою.

Для того, щоб почати користуватися сервісом необхідно мати лише встановлений браузер та доступ до глобальної мережі.

Додаток 3

Система обліку надзвичайних ситуацій
природного характеру

Текст програмного модулю

УКР.НТУУ”КПІ”_ТЕФ_АПЕПС_ТМ62

Аркушів 26

Київ 2020

EmergenciesController.cs

```
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
using System.Threading.Tasks;
using Diplom.DataModels;
using Microsoft.AspNetCore.Http;
using Diplom.Services;
using Diplom.ViewModels;

namespace Diplom.Controllers
{
   [ApiController]
   [Route("api/[controller]")]
   public class EmergenciesController : ControllerBase
   {
      private readonly IEmergencyService _emergencyService;

      public EmergenciesController(IEmergencyService emergencyService)
      {
         _emergencyService = emergencyService;
      }

      [HttpGet]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Emergency>> GetAsync()
      {
         return await _emergencyService.ListAsync();
      }


      [HttpGet("{emergencyId}")]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Emergency>> GetAsync(int emergencyId)
      {
         return await _emergencyService.ListAsync(emergencyId);
      }

      [HttpGet("statistic")]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<StatisticViewModel> GetStatisticAsync()
      {
         return await _emergencyService.GetStatisticAsync();
      }
   }
}
```

EventsController.cs

```
using System.Collections.Generic;
using Microsoft.AspNetCore.Mvc;
```

```
using System.Threading.Tasks;
using Diplom.DataModels;
using Microsoft.AspNetCore.Http;
using Diplom.Services;
using System;
using Microsoft.AspNetCore.Authorization;
using Diplom.ViewModels;
using System.Text.RegularExpressions;
using Newtonsoft.Json.Linq;
using Newtonsoft.Json;

namespace Diplom.Controllers
{
   [ApiController]
   [Route("api/[controller]")]
   public class EventsController : ControllerBase
   {
      private readonly IEventService _eventService;

      public EventsController(IEventService eventService)
      {
         _eventService = eventService;
      }

      [HttpGet]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Event>> GetAsync()
      {
         return await _eventService.ListAsync();
      }


      [HttpGet("{eventId}")]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Event>> GetAsync(int eventId)
      {
         return await _eventService.ListAsync(eventId);
      }

      [HttpGet("dates/{date}")]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Event>> GetAsync(DateTime date)
      {
         return await _eventService.ListAsync(date);
      }

      [HttpGet("{date}/{emergencyId}")]
      [ProducesResponseType(StatusCodes.Status200OK)]
      public async Task<IEnumerable<Event>> GetAsync(DateTime date, int emergencyId)
```

```
    {
        return await _eventService.ListAsync(date, emergencyId);
    }

    [HttpGet("dates")]
    [ProducesResponseType(StatusCodes.Status200OK)]
    public async Task<List<DateTime?>> GetDatesAsync()
    {
        return await _eventService.DatesListAsync();
    }

    [HttpPut("update")]
    [ProducesResponseType(StatusCodes.Status200OK)]
    public async Task<ActionResult> UpdateEventAsync([FromBody] JObject jEventDTO)
    {
        JsonSerializer serializer = new JsonSerializer();
        EventDTO eventDTO = (EventDTO)serializer.Deserialize(new JTokenReader(jEventDTO),
typeof(EventDTO));
        try
        {
            var result = await _eventService.UpdateEventAsync(eventDTO);

            if (!result.Success)
                return BadRequest(result.Message);

        }
        catch (Exception e)
        {
            return BadRequest(e.Message);
        }

        return Ok();
    }

    // POST: api/Event/add
    [HttpPost("add"), DisableRequestSizeLimit]
    [ProducesResponseType(StatusCodes.Status200OK)]
    public async Task<ActionResult> AddEventAsync([FromBody] JObject jEventDTO)
    {
        JsonSerializer serializer = new JsonSerializer();
        EventDTO eventDTO = (EventDTO)serializer.Deserialize(new JTokenReader(jEventDTO),
typeof(EventDTO));
        try
        {
            var result = await _eventService.AddEventAsync(eventDTO);

            if (!result.Success)
                return BadRequest(result.Message);
```

```
            }
            catch (Exception e)
            {
                return BadRequest(e.Message);
            }

            return Ok();
        }

        // POST: api/Event/delete
        [HttpPost("delete"), ActionName("Delete")]
        public async Task<IActionResult> Delete(int eventId)
        {
            var result = await _eventService.DeleteAsync(eventId);

            if (!result.Success)
                return BadRequest(result.Message);

            return Ok();
        }
    }
}
```

DiplomDatabaseContext.cs

```
using System;
using Microsoft.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore.Metadata;
using Diplom.DataModels;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

namespace Diplom.Data
{
    public class DiplomDatabaseContext : IdentityDbContext<AppUser>
    {
        public DiplomDatabaseContext()
        { }

        public DiplomDatabaseContext(DbContextOptions<DiplomDatabaseContext> options)
            : base(options)
        { }

        public virtual DbSet<AppUserInfo> AppUserInfos { get; set; }
        public virtual DbSet<Emergency> Emergencies { get; set; }
        public virtual DbSet<Event> Events { get; set; }
        public virtual DbSet<EventPosition> EventPositions { get; set; }

        protected override void OnModelCreating(ModelBuilder builder)
        {
            base.OnModelCreating(builder);
```

```
            builder.Entity<Event>(b =>
            {
                b.HasKey(e1 => e1.EventId);
                b.OwnsOne(e1 => e1.EventPosition, md => {
                    md.ToTable("EventPosition");
                });

                b.ToTable("Event");
            });


        }
    }
}
```

Emergency.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Diplom.DataModels
{
    public class Emergency
    {
        public int EmergencyId { get; set; }
        public string Name { get; set; }

        public ICollection<Event> Events { get; set; }
    }
}
```

Event.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Diplom.DataModels
{
    public class Event
    {
        public int EventId { get; set; }
        public string EventName { get; set; }
        public string Description { get; set; }
        public int? Harmed { get; set; }
        public int? Deaths { get; set; }
        public double? Losses { get; set; }
```

```
        public double? Costs { get; set; }
        public DateTime? Date { get; set; }
        public string ImageData { get; set; }

        public EventPosition EventPosition { get; set; }

        public int EmergencyId { get; set; }
        public Emergency Emergency { get; set; }
    }
}
```

EventPosition.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;

namespace Diplom.DataModels
{
    public class EventPosition
    {
        public Event Event { get; set; }
        public double X { get; set; }
        public double Y { get; set; }
        public string Place { get; set; }
    }
}
```

EventDTO.cs

```
using Diplom.DataModels;
using Diplom.ViewModels.Validations;
using Microsoft.AspNetCore.Http;
using ServiceStack.FluentValidation.Attributes;
using System;
using System.Collections.Generic;

namespace Diplom.ViewModels
{
    public class EventDTO
    {
        public int EventId { get; set; }
        public string EventName { get; set; }
        public string Description { get; set; }
        public int? Harmed { get; set; }
        public int? Deaths { get; set; }
        public double? Losses { get; set; }
        public double? Costs { get; set; }
```

```
        public DateTime? Date { get; set; }
        public List<string> ImageByteArrayList { get; set; }

        public EventPosition EventPosition { get; set; }

        public int EmergencyId { get; set; }
        public Emergency Emergency { get; set; }
    }
}
```

CredentialsViewModelValidator.cs

```
using FluentValidation;

namespace Diplom.ViewModels.Validations
{
    public class CredentialsViewModelValidator : AbstractValidator<CredentialsViewModel>
    {
        public CredentialsViewModelValidator()
        {
            RuleFor(vm => vm.UserName).NotEmpty().WithMessage("Username cannot be empty");
            RuleFor(vm => vm.Password).NotEmpty().WithMessage("Password cannot be empty");
            RuleFor(vm => vm.Password).Length(6, 12).WithMessage("Password must be between 6
and 12 characters");
        }
    }
}
```

AddEventComponent.ts

```
import { Component, OnInit, Input } from '@angular/core';

import { Marker } from '../shared/models/Marker';
import { Event } from '../shared/models/event';
import {NgbDateParserFormatter, NgbDate} from '@ng-bootstrap/ng-bootstrap';
import { EMERGENCY_FE_NAME_TO_ID, EMERGENCY_FE_ID_TA_NAME } from
'../shared/models/emergency';
import { EventService } from '../shared/services/event.service';
import { ActivatedRoute, Router } from '@angular/router';
import { map } from 'rxjs/operators'
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-add-event',
  templateUrl: './add-event.component.html',
  styleUrls: ['./add-event.component.scss']
})
export class AddEventComponent implements OnInit {
```

```
  constructor( private ngbDateParserFormatter: NgbDateParserFormatter,
    private eventService: EventService,
    private activatedRoute: ActivatedRoute,
    private router: Router,
    private http: HttpClient) { }
    isExistedEvent: boolean = false;

  newMarker: Marker;
  currentEvent: Event;
  latitude: number;
  longitude: number;
  zoom: number;
  currentEventDate: NgbDate;
  emergencyType: string;
  selectedFiles: {name: string, data: string}[] = []

  ngOnInit() {
    this.newMarker = new Marker();
    this.newMarker = { lat: 0, lng: 0,name: "", address: "", desc: "", eventId: null };
    this.currentEvent = new Event();

    // this.currentEventDate = new Date('2020');
    // console.log(this.currentEventDate.toISOString(), this.currentEventDate)
    this.latitude = 49.045639;
    this.longitude = 31.159608;
    this.zoom = 5.2;
    this.activatedRoute.paramMap
      .pipe(map(() => window.history.state))
      if(window.history.state.existedEvent){
        this.setCurrentEvent(window.history.state.existedEvent);
        this.isExistedEvent = true
      }
      this.currentEvent.imageByteArrayList = this.currentEvent.imageByteArrayList || [];
  }

  setCurrentEvent(event: Event){
    this.currentEvent = event;
    let date = new Date(event.date)
    this.currentEventDate = new NgbDate(date.getFullYear(), date.getMonth(), date.getDay());
    this.newMarker = <Marker>{ lat: event.eventPosition.x, lng: event.eventPosition.y, name:
event.eventName, address: event.eventPosition.place, desc: event.description, eventId:
event.eventId }
    this.emergencyType = EMERGENCY_FE_ID_TA_NAME.get(event.emergencyId).toLowerCase()
  }

  addMarker(lat: number, lng: number) {
    this.newMarker.lat = lat;
    this.newMarker.lng = lng;
    console.log(this.newMarker)
```

```
  }

  selectedFile: File

  onFileChanged(event) {
    this.selectedFile = event.target.files[0];
    console.log(event)
    this.onUpload()
  }

  onUpload() {
    console.log(this.selectedFile)

    var reader = new FileReader();
    let thiss = this
    reader.onload = function () {
      console.log(reader.result);
      thiss.selectedFiles.push({name: thiss.selectedFile.name, data: <string> reader.result})
    }

    reader.readAsDataURL(this.selectedFile);
  }
  removeSelectedFile(file){
    const index = this.selectedFiles.indexOf(file);
    if(index > -1){
      this.selectedFiles.splice(index, 1);
    }
  }


  onSubmited() {
    this.currentEvent.harmed = (this.currentEvent.harmed && this.currentEvent.harmed > 0 ) ?
this.currentEvent.harmed : 0;
    this.currentEvent.deaths = (this.currentEvent.deaths && this.currentEvent.deaths > 0 ) ?
this.currentEvent.deaths : 0;
    this.currentEvent.losses = (this.currentEvent.losses && this.currentEvent.losses > 0 ) ?
this.currentEvent.losses : 0;
    this.currentEvent.costs = (this.currentEvent.costs && this.currentEvent.costs > 0 ) ?
this.currentEvent.costs : 0;

    this.currentEvent.date = new
Date(this.ngbDateParserFormatter.format(this.currentEventDate)).toISOString();
    this.currentEvent.emergencyId =
EMERGENCY_FE_NAME_TO_ID.get(this.emergencyType.toUpperCase());

    this.currentEvent.eventPosition.x = this.newMarker.lat
    this.currentEvent.eventPosition.y = this.newMarker.lng
    console.log(this.newMarker, this.currentEvent);
    this.currentEvent.imageByteArrayList = this.selectedFiles.map(file => file.data)
```

```
    console.log(this.currentEvent.imageByteArrayList)
    this.currentEvent.emergency = null;

    this.eventService.modifyEvent(this.currentEvent, this.isExistedEvent);
    // this.router.navigate(["/map"])
  }
}
```

AddEventComponent.html

```
<div class="col-md-6 login-form-center">
  <div class="card">
    <form (ngSubmit)="onSubmited()" #newMarkerForm="ngForm">

      <div class="form-group">
        <label for="name">Назва НС <span style="color: red;">*</span></label>
        <input type="text" class="form-control" id="name" required
[(ngModel)]="currentEvent.eventName"
          name="name" #name="ngModel">
        <div [hidden]="name.valid || name.pristine" class="alert alert-danger">
          Name is required
        </div>
      </div>

      <div class="form-group">
        <label for="name">Місце НС <span style="color: red;">*</span></label>
        <input type="text" class="form-control" id="address" required
          [(ngModel)]="currentEvent.eventPosition.place" name="address"
#address="ngModel">
        <div [hidden]="address.valid || address.pristine" class="alert alert-danger">
          Address is required
        </div>
      </div>

      <div class="form-group">
        <label for="name">Опис ситуації <span style="color: red;">*</span></label>
        <input type="text" class="form-control" id="desc" required
[(ngModel)]="currentEvent.description"
          name="desc" #desc="ngModel">
        <div [hidden]="desc.valid || desc.pristine" class="alert alert-danger">
          Description is required
        </div>
      </div>

      <div class="form-group">
        <label for="name">Кількість постраждалих осіб</label>
        <input type="number" min="0" class="form-control" id="harmed"
[(ngModel)]="currentEvent.harmed"
          name="harmed" #harmed="ngModel">
      </div>
```

```
        <div class="form-group">
          <label for="name">Кількість померлих осіб</label>
          <input type="number" min="0" class="form-control" id="deaths"
[(ngModel)]="currentEvent.deaths"
            name="deaths" #deaths="ngModel">
        </div>
        <div class="form-group">
          <label for="name">Збитки</label>
          <input type="number" min="0" class="form-control" id="losses"
[(ngModel)]="currentEvent.losses"
            name="losses" #losses="ngModel">
        </div>
        <div class="form-group">
          <label for="name">Вартість ліквідації</label>
          <input type="number" min="0" class="form-control" id="costs"
[(ngModel)]="currentEvent.costs"
            name="costs" #costs="ngModel">
        </div>
        <div class="form-group">
          <label for="name">Дата НС <span style="color: red;">*</span></label>

          <div class="input-group">
            <input class="form-control" id="date" placeholder="yyyy-mm-dd" name="date"
              [(ngModel)]="currentEventDate" required ngbDatepicker #date="ngbDatepicker">
            <div class="input-group-append">
              <button class="btn btn-outline-secondary calendar" (click)="date.toggle()"
                type="button"></button>
            </div>
          </div>
        </div>
        <div class="form-group">
          <label for="name">Характер НС <span style="color: red;">*</span></label>

          <select class="form-control" id="emergency_name" required
[(ngModel)]="emergencyType"
            name="emergency_name" #emergency_name="ngModel">
            <option>техногенного</option>
            <option>природного</option>
            <option>соціально-політичного</option>
            <option>воєнного</option>
          </select>

          <div [hidden]="address.valid || address.pristine" class="alert alert-danger">
            Address is required
          </div>
        </div>

        <div class="form-group">
          <label for="name">Місце НС на карті <span style="color: red;">*</span></label>
```

```
            <agm-map class="marker-map" [latitude]="latitude" [longitude]="longitude"
[zoom]="zoom"
              (mapClick)="addMarker($event.coords.lat, $event.coords.lng)">
              <agm-marker [latitude]="newMarker.lat" [longitude]="newMarker.lng"></agm-
marker>
            </agm-map>
          </div>
          <input style="display: none" type="file" accept="image/png"
(change)="onFileChanged($event)" #fileInput>
          <button type="button" class="btn btn-secondary" (click)="fileInput.click()">Select
File</button>
          <ng-container *ngFor="let file of selectedFiles">
            <p> {{file.name}} <span (click)=removeSelectedFile(file)> x </span></p>
          </ng-container>
          <button type="submit" class="form__button"
            [disabled]="!newMarkerForm.form.valid || newMarker.lat === 0">Submit</button>

        </form>
      </div>
    </div>
```

CalcLossFireComponent.ts

```
import { Component, OnInit, Input } from '@angular/core';
import { Marker } from '../shared/models/Marker';
import { Event } from '../shared/models/event';
import {NgbDateParserFormatter, NgbDate} from '@ng-bootstrap/ng-bootstrap';
import { EMERGENCY_FE_NAME_TO_ID, EMERGENCY_FE_ID_TA_NAME } from
'../shared/models/emergency';
import { EventService } from '../shared/services/event.service';
import { ActivatedRoute, Router } from '@angular/router';
import { map } from 'rxjs/operators'
import { HttpClient } from '@angular/common/http';

@Component({
  selector: 'app-calc-loss-fire',
  templateUrl: './calc-loss_fire.component.html',
  styleUrls: ['./calc-loss_fire.component.scss']
})
export class CalcLossFireComponent implements OnInit {

  inputParams: {
    P: number;
    obl: number;
    groupBefore: number;
    groupAfter: number;
    koefProd: number;
    koefLostProd: number;
  } = {
```

```
  P: null,
  obl: null,
  groupBefore: null,
  groupAfter: null,
  koefProd: null,
  koefLostProd: null
 }

 result: {
  Plr1: number;
  Plr2: number;
  Plr3: number;
  Plr: number;
 } = {
  Plr1: 0,
  Plr2: 0,
  Plr3: 0,
  Plr: 0
 }

 constructor() { }

 ngOnInit() {  }

 onSubmited() {
console.log(this.inputParams)

  this.result.Plr1 = this.getNormatives(this.inputParams.obl, this.inputParams.groupBefore) *
this.inputParams.koefProd * this.inputParams.P;
  this.result.Plr2 = (1 - this.inputParams.koefLostProd) *
this.getNormatives(this.inputParams.obl, this.inputParams.groupBefore)  * this.inputParams.P;
  this.result.Plr3 = (this.getNormatives(this.inputParams.obl, this.inputParams.groupBefore) -
this.getNormatives(this.inputParams.obl, this.inputParams.groupAfter)) *
this.inputParams.koefProd * this.inputParams.P;
  this.result.Plr = this.result.Plr1 + this.result.Plr2 + this.result.Plr3;
  console.log(this.getNormatives(27, 1))
 }

 getNormatives( i: number, j: number): number{
  let normatives: number[][] =
  [[84.7,        50.2],
   [77.7,        46],
   [145          , 0],
   [163          , 0],
   [75,44.4],
   [29.8,        17.9],
   [123.1,       123.1],
   [250.9, 0],
   [31.9,        19.9],
```

```
    [123.1,      123.1],
    [80.5,       47.7],
    [159.1,      94.3],
    [118.6, 0    ],
    [70.1,       41.6],
    [123.1,      123.1],
    [241.6, 0    ],
    [141.8,      0],
    [135, 0      ],
    [74.1,       43.9],
    [19.6,       47.1],
    [100.4,      59.5],
    [91.9, 0     ],
    [167.3, 0    ],
    [93.2,       55.2],
    [75.8,       44.9],
    [31.1,       18.7],
    [123.1,      123.1],
    [75,44.4]]

return normatives[i][j]
  }
}
```

CalcLossFireComponent.html

```
<div class="col-md-8 login-form-center">
  <div class="card">
    <form (ngSubmit)="onSubmited()" #newMarkerForm="ngForm">
      <h3>Розрахунок збитків від втрати деревини та інших лісових ресурсів </h3>
      <div class="calc-block">
        <div class="form-group">
          <label for="name">Площа лісової ділянки, що вилучається або знищується, у
гектарах</label>
          <input type="text" class="form-control" id="name" [(ngModel)]="inputParams.P"
name="name"
            #name="ngModel">
        </div>
        <div class="form-group">
          <label for="name">Область <span style="color: red;">*</span> </label>
          <select class="form-control" id="emergency_name" required
[(ngModel)]="inputParams.obl"
            name="emergency_name" #emergency_name="ngModel">
            <option value="0">Вінницька</option>
            <option value="1">Волинська</option>
            <option value="2">Дніпропетровська</option>
            <option value="3">Донецька</option>
            <option value="4">Житомирська</option>
            <option value="5">Закарпатська </option>
            <option value="6">Закарпатська гірська частина області</option>
```

```
                <option value="7">Запорізька</option>
                <option value="8">Івано-Франківська</option>
                <option value="9">гірська частина області</option>
                <option value="10">Київська </option>
                <option value="11">Кіровоградська</option>
                <option value="12">Луганська</option>
                <option value="13">Львівська </option>
                <option value="14">Львівська гірська частина області</option>
                <option value="15">Миколаївська</option>
                <option value="16">Одеська</option>
                <option value="17">Полтавська</option>
                <option value="18">Рівненська </option>
                <option value="19">Сумська</option>
                <option value="20">Тернопільська</option>
                <option value="21"> Харківська</option>
                <option value="22">Херсонська</option>
                <option value="23">Хмельницька</option>
                <option value="24">Черкаська</option>
                <option value="25">Чернівецька</option>
                <option value="26">Чернівецька гірська частина області</option>
                <option value="27">Чернігівська </option>

              </select>
            </div>
            <div class="form-group">
              <label for="name">Група лісу до НС <span style="color: red;">*</span> </label>
              <select class="form-control" id="emergency_name" required
[(ngModel)]="inputParams.groupBefore"
                name="emergency_name" #emergency_name="ngModel">
                <option value="0">1</option>
                <option value="1">2</option>
              </select>
            </div>

            <div class="form-group">
              <label for="name">Група лісу після НС <span style="color: red;">*</span> </label>
              <select class="form-control" id="emergency_name" required
[(ngModel)]="inputParams.groupAfter"
                name="emergency_name" #emergency_name="ngModel">
                <option value="0">1</option>
                <option value="1">2</option>
              </select>
            </div>
            <div class="form-group">
              <label for="name">Коефіцієнт продуктивності лісів</label>
              <input type="text" class="form-control" id="name"
[(ngModel)]="inputParams.koefProd" name="name"
                #name="ngModel">
            </div>
```

```
          <div class="form-group">
            <label for="name">Коефіцієнт зниження продуктивності угіддя</label>
            <input type="text" class="form-control" id="name"
[(ngModel)]="inputParams.koefLostProd" name="name"
              #name="ngModel">
          </div>
          <h3>{{result.Plr}} тис. грн.</h3>
        </div>
        <button type="submit" class="form__button">Підрахувати</button>

      </form>
      <br>
      <br>
      <img src="assets/images/koefProd.png">
    </div>
</div>
```

MainMapComponent.html

```
<div class="map-page">
  <div class="main-block">
    <div class="row map-row">
      <div class="col-8">
        <agm-map class="main-map" [latitude]="latitude" [longitude]="longitude"
[zoom]="zoom"
          [backgroundColor]="green">
          <agm-marker *ngFor="let marker of markers" [latitude]="marker.lat"
[longitude]="marker.lng"
            (markerClick)="selectMarker(marker)"></agm-marker>
        </agm-map>
      </div>
      <div class="col-4 marker-info">
        <div *ngIf="currentMarker; else no_marker" >
          <h1>{{ currentMarker.name }}</h1>
          <h5><i>{{ currentMarker.address }}</i></h5>
          {{ currentMarker.desc }}
          <div *ngIf="status" class="marker-info__edit-icon">
            <svg (click)="updateEvent()" width="25" height="25" viewBox="0 0 25 25"
fill="none" xmlns="http://www.w3.org/2000/svg"
              xmlns:xlink="http://www.w3.org/1999/xlink">
              <rect width="25" height="25" fill="url(#pattern0)" fill-opacity="0.3" />
              <defs>
                <pattern id="pattern0" patternContentUnits="objectBoundingBox" width="1"
height="1">
                  <use xlink:href="#image0" transform="scale(0.00195312)" />
                </pattern>
              </defs>
            </svg>
          </div>
```

```
          </div>
          <ng-template #no_marker>

            <h1>Виберіть, будь ласка, точку на карті</h1>
            <h5><i>Тут буде виведено інформацію про подію</i></h5>
          </ng-template>
        </div>
      </div>
    </div>

    <div *ngIf="currentMarker && currentMarker.slides && currentMarker.slides.length > 0"
class="slider-block">
      <div class="slider-block__init row justify-content-around">
        <!-- <ngx-slick-carousel class="carousel" #slickModal="slick-carousel"
[config]="slideConfig"> -->
          <div  *ngFor="let slide of currentMarker.slides" class="slide">
            <img src="{{ slide.img }}" alt="no emergency photo" width="100%">
          </div>
        <!-- </ngx-slick-carousel> -->
      </div>
    </div>
  </div>
```

MainMapComponent.ts

```
import { Component, OnInit, Input, OnDestroy } from '@angular/core';
import { Marker } from '../shared/models/Marker';
import { EventService } from '../shared/services/event.service';
import { Event } from '../shared/models/event';
import { Observable, Subscription, of } from 'rxjs';
import { UserService } from '../shared/services/user.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-main-map',
  templateUrl: './main-map.component.html',
  styleUrls: ['./main-map.component.scss']
})
export class MainMapComponent implements OnInit, OnDestroy {

  constructor(private eventService: EventService,
    private userService: UserService,
    private router: Router) { }


  status: boolean;
  subscription: Subscription;
  latitude: number;
  longitude: number;
```

```
 zoom: number;
 markers: Marker[];
 currentMarker: Marker;
 slides: { img: string }[][] = [];
  events: Event[]
 events$: Observable<Event[]>;
 @Input()
 ngOnInit() {
  this.markers = [];
  this.setInit();

  this.subscription = this.userService.authNavStatus$.subscribe(status => this.status = status);
 }

 ngOnDestroy() {
  this.subscription.unsubscribe();
 }

 // Get Current Location Coordinates
 private setInit() {
  this.latitude = 49.045639;
  this.longitude = 31.159608;
  this.zoom = 6;
  this.setInitMarkers();
 }

 updateEvent(){
  let currentEvent = this.events.find(event => event.eventId == this.currentMarker.eventId);
  this.router.navigate(['/add-event'], {state: {existedEvent: currentEvent}});
 }

 private setInitMarkers() {
  this.eventService.events.subscribe(events => {
   this.events = events;
   this.markers = events.filter(event => event.eventPosition).map((event) => {
    let slides = [];
    if(event.imageData){
     const images = event.imageData.split("|");
     console.log(images);
     images.forEach(img => slides.push({img: img}))
    }
     return <Marker>{ lat: event.eventPosition.x, lng: event.eventPosition.y, name:
event.eventName, address: event.eventPosition.place, desc: event.description, eventId:
event.eventId, slides: slides } });
   console.log(this.markers, this.events)
  });
 }

 selectMarker(marker: Marker) {
```

```
    this.currentMarker = marker;
  }
}
```

TablesPageComponent.html

```
<div class="table-block">
  <div class="table-block__title"> Довідкові дані щодо кількості класифікованих ситуацій з початку року:</div>
  <div class=" table-block__table">
    <div class="row">
      <div class="col-1 table-item __header"> Всього НС</div>
      <div class="col-7">
        <div class="row table-item __header"> Характер НС</div>
        <div class="row">
          <div class="col-3 table-item __header">техногенний</div>
          <div class="col-3 table-item __header">природний</div>
          <div class="col-3 table-item __header">соціальний</div>
          <div class="col-3 table-item __header">військовий</div>
        </div>
      </div>
      <div class="col-2 table-item __header">Загинуло, осіб</div>
      <div class="col-2 table-item __header">Постраждало, осіб</div>
    </div>
    <ng-container *ngIf="statistic$ | async as statistic">

      <div class="row">
        <div class="col-1 table-item __content">{{statistic.eventsNumber}}</div>
        <div class="col-7">
          <div class="row">
            <div class="col-3 table-item __content">{{statistic.technogenicNumber}}</div>
            <div class="col-3 table-item __content">{{statistic.naturalNumber}}</div>
            <div class="col-3 table-item __content">{{statistic.socialNumber}}</div>
            <div class="col-3 table-item __content">{{statistic.militaryNumber}}</div>
          </div>
        </div>
        <div class="col-2 table-item __content">{{statistic.deathsCount}}</div>
        <div class="col-2 table-item __content">{{statistic.harmedCount}}</div>
      </div>
    </ng-container>
  </div>
</div>
<div class="table-block">
  <div class="table-block__title">
    <div class="row" style="width: 100%;">
      <div class="col-5">Надзвичайні ситуації:</div>
      <div class="col-4"> Характер НС:
```

```
            <select (change)="selectorsService.typeChanged($event)"
              class="main-block__options-panel_select-btn __type">
              <option>ВСІ</option>
              <option [selected]="selectorsService.currentEmergencyTypeId ==
1">техногенного</option>
              <option [selected]="selectorsService.currentEmergencyTypeId ==
2">природного</option>
              <option [selected]="selectorsService.currentEmergencyTypeId == 3">соціально-
політичного</option>
              <option [selected]="selectorsService.currentEmergencyTypeId ==
4">воєнного</option>
            </select>
          </div>
          <div class="col-3">
            Рік:
            <select (change)="selectorsService.yearChanged($event)"
              class="main-block__options-panel_select-btn __year">
              <option>всі</option>
              <ng-container *ngIf="selectorsService.dates$ | async as dates">
                <ng-container *ngFor="let date of dates">
                  <option [selected]="selectorsService.currentEventYear ==
date">{{date}}</option>
                </ng-container>
              </ng-container>
            </select>
          </div>

      </div>
    </div>
    <div class=" table-block__table">
      <div class="row" [ngClass]="{'table-block__table-header': status}">
        <div class="col-3 table-item __header"> Назва НС</div>
        <div class="col-2 table-item __header"> Дата</div>
        <div class="col-3 table-item __header"> Місце, де виникла НС</div>
        <div class="col-2 table-item __header"> Збитки</div>
        <div class="col-2 table-item __header"> Вартість ліквідації</div>
      </div>
      <ng-container *ngIf="eventService.events | async as events">
        <ng-container *ngFor="let event of events">
          <div class="table-block__table_row">
          <div class="row table-block__table_row_elements">
            <div class="col-3 table-item __content">{{event.eventName}}</div>
            <div class="col-2 table-item __content">{{event.date | date}}</div>
            <div class="col-3 table-item __content">{{event.eventPosition.place || ""}}</div>
            <div class="col-2 table-item __content">{{event.losses || 0}}</div>
            <div class="col-2 table-item __content">{{event.costs || 0}}</div>
          </div >
          <div *ngIf="status" class="table-block__table_edit-icon">
```

```
            <svg (click)="updateEvent(event)" width="25" height="25" viewBox="0 0 25 25"
fill="none"
                xmlns="http://www.w3.org/2000/svg"
xmlns:xlink="http://www.w3.org/1999/xlink">
                <rect width="25" height="25" fill="url(#pattern0)" fill-opacity="0.3" />
                <defs>
                  <pattern id="pattern0" patternContentUnits="objectBoundingBox" width="1"
height="1">
                    <use xlink:href="#image0" transform="scale(0.00195312)" />
                  </pattern>
                  </defs>
              </svg>
            </div>
          </div>
          </ng-container>
      </ng-container>
      <div *ngIf="status" class="table-block__table_edit-icon">
        Додати НС
        <svg (click)="updateEvent(event)" width="25" height="25" viewBox="0 0 25 25"
fill="none"
          xmlns="http://www.w3.org/2000/svg" xmlns:xlink="http://www.w3.org/1999/xlink">
          <rect width="25" height="25" fill="url(#pattern0)" fill-opacity="0.3" />
          <defs>
            <pattern id="pattern0" patternContentUnits="objectBoundingBox" width="1"
height="1">
              <use xlink:href="#image0" transform="scale(0.00195312)" />
            </pattern>
          </defs>
        </svg>
      </div>
    </div>
</div>
```

TablesPageComponent.ts

```
import { Component, OnInit } from '@angular/core';
import { EventService } from 'src/app/shared/services/event.service';
import { EmergencyService } from 'src/app/shared/services/emergency.service';
import { Observable, Subscription } from 'rxjs';
import { SelectorsService } from 'src/app/shared/services/selectors.service';
import { UserService } from 'src/app/shared/services/user.service';
import { Router } from '@angular/router';

@Component({
  selector: 'app-tables-page',
  templateUrl: './tables-page.component.html',
  styleUrls: ['./tables-page.component.scss']
})
```

```
export class TablesPageComponent implements OnInit {

  constructor(private eventService: EventService,
    private userService: UserService,
    private emergencyService: EmergencyService,
    private selectorsService: SelectorsService,
    private router: Router) { }

  events$: Observable<any>;

  status: boolean;
  subscription: Subscription;
  statistic$: Observable<any>
  ngOnInit() {
    this.subscription = this.userService.authNavStatus$.subscribe(status => this.status = status);
    this.statistic$ = this.emergencyService.getEmergenciesStatistic();
  }

  show(date) {
    console.log(date)
  }

  updateEvent(event: Event){
    this.router.navigate(['/add-event'], {state: {existedEvent: event}});
  }

}
```

EmergencyService.ts

```
import { throwError, Observable, BehaviorSubject } from 'rxjs';
import { ConfigService } from '../utils/config.service';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';

@Injectable()

export class EmergencyService {

    baseUrl: string;

    constructor(private configService: ConfigService,
        private http: HttpClient) {
        this.baseUrl = this.configService.getApiURI();
    }

     getEmergenciesStatistic():Observable<any>{
         return this.http.get<any>(this.baseUrl + "/Emergencies/statistic")
```

```
    }
}
```

EventService.ts

```
import { throwError, Observable, BehaviorSubject } from 'rxjs';
import { ConfigService } from '../utils/config.service';
import { HttpClient, HttpHeaders } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { Event } from '../models/event';
import { Emergency } from '../models/emergency';
import { map } from 'rxjs/operators'
import { Router } from '@angular/router';

@Injectable()

export class EventService {

  baseUrl: string;
  events: BehaviorSubject<Event[]> = new BehaviorSubject([])
  currentEvents = this.events.asObservable();

  constructor(private configService: ConfigService,
    private http: HttpClient,
    private router: Router) {
    this.baseUrl = this.configService.getApiURI();
    this.getAllEvents();
  }

  getAllEvents() {
    this.http.get<Event[]>(this.baseUrl + "/Events").subscribe(data => this.events.next(data))
  }

  getEventsByEmergency(id: number) {
    this.http.get<Emergency[]>(this.baseUrl + "/Emergencies/" + id).subscribe(data => {
      let events: Event[] = [];
      data.forEach(emergency => events.push(...emergency.events))
      this.events.next(events)
    })
  }

  getEventsByYear(date: Date) {
    this.http.get<Event[]>(this.baseUrl + "/Events/dates/" + date.toISOString()).subscribe(data
=> this.events.next(data));
  }

  getEventsByYearAndEmergency(date: Date, id: number) {
    this.http.get<Event[]>(this.baseUrl + `/Events/${date.toISOString()}/${id}`).subscribe(data
=> this.events.next(data));
```

```
  }


  getAvailableDate(): Observable<Date[]> {
    return this.http.get<Date[]>(this.baseUrl + "/Events/dates");
  }

  modifyEvent(event: Event, isExist: boolean): void {
    console.log(event)

    let headers = new HttpHeaders();
    headers = headers.append('Content-Type', 'application/json');
    let authToken = localStorage.getItem('auth_token');
    headers = headers.set('Authorization', `Bearer ${authToken}`);

    if(isExist){
      this.http.put(this.baseUrl + "/Events/update", event, {headers} )
      .subscribe(data => {
        console.log(data)
        this.router.navigate(["/map"])
        this.getAllEvents();
      },
      error => alert(error.error));
    } else{
      this.http.post(this.baseUrl + "/Events/add", event, {headers}).pipe(map((response:
Response) => console.log(response)))
      .subscribe(data => {
        console.log(data)
        this.router.navigate(["/map"])
        this.getAllEvents();
      },
      error => alert(error.error));

    }
  }

}
```

SelectorsService.ts

```
import { EventService } from "./event.service";
import { Injectable } from "@angular/core";
import { Observable, of } from "rxjs";
import { map } from "rxjs/operators";

import { EMERGENCY_FE_NAME_TO_ID } from "../models/emergency";

@Injectable()
```

```
export class SelectorsService {

 currentEmergencyTypeId: number;
 currentEventYear: string;
 dates$: Observable<string[]>;

  constructor(private eventService: EventService) {
  this.dates$ = this.eventService.getAvailableDate()
        .pipe(map( dates =>  dates.map(date => new Date(date).getFullYear().toString())));
  }

  typeChanged($event) {

    if ($event.target.value == "BCI") {
      this.currentEmergencyTypeId = null;
    } else {
      this.currentEmergencyTypeId =
EMERGENCY_FE_NAME_TO_ID.get($event.target.value.toUpperCase())
    }
    console.log(this.currentEmergencyTypeId);
    this.getEvents();
  }

  yearChanged($event) {
    if ($event.target.value == "BCI" || $event.target.value == "вci") {
      this.currentEventYear = null;
    } else {
      this.currentEventYear = $event.target.value
    }
    console.log(this.currentEventYear, $event.target.value);
    this.getEvents();

  }

  getEvents() {
    console.log(this.currentEmergencyTypeId, this.currentEventYear)
    if (this.currentEmergencyTypeId && this.currentEventYear) {
      this.eventService.getEventsByYearAndEmergency(new Date(this.currentEventYear),
this.currentEmergencyTypeId);
    } else if (this.currentEmergencyTypeId) {
      this.eventService.getEventsByEmergency(this.currentEmergencyTypeId);
    } else if (this.currentEventYear) {
      this.eventService.getEventsByYear(new Date(this.currentEventYear));
    } else {
      this.eventService.getAllEvents();
    }
  }
}
```

UserService.ts

```
import { Injectable } from '@angular/core';
import { HttpClient, HttpResponse, HttpHeaders } from '@angular/common/http';

import { UserRegistration } from '../models/user.registration.interface';
import { ConfigService } from '../utils/config.service';

import {BaseService} from "./base.service";

import { Observable, BehaviorSubject } from 'rxjs';
import { map, catchError } from 'rxjs/operators';

@Injectable()

export class UserService extends BaseService {

  baseUrl: string = '';

  // Observable navItem source
  private _authNavStatusSource = new BehaviorSubject<boolean>(false);
  // Observable navItem stream
  authNavStatus$ = this._authNavStatusSource.asObservable();

  private loggedIn = false;
  httpOptions = {
    headers: new HttpHeaders({ 'Content-Type': 'application/json' })
  };
  constructor(private http: HttpClient, private configService: ConfigService) {
    super();
    this.loggedIn = !!localStorage.getItem('auth_token');
    this._authNavStatusSource.next(this.loggedIn);
    this.baseUrl = configService.getApiURI();
  }


   login(userName, password) {
    console.log(this.baseUrl + '/auth/login', JSON.parse(JSON.stringify({ userName, password })),
this.httpOptions)

    return this.http
      .post(
      this.baseUrl + '/auth/login',
      JSON.stringify({ userName, password }), this.httpOptions
      )
      .pipe(map(res => { console.log(res); return JSON.parse( JSON.stringify(res))}),
      map(res => {
```

```
      localStorage.setItem('auth_token', res.auth_token);
      localStorage.setItem('user_name', res.user_name);
      console.log(res, localStorage);

      this.loggedIn = true;
      this._authNavStatusSource.next(true);
      return true;
    }),
    catchError(this.handleError)
    );
  return null;
 }

 logout() {
  localStorage.removeItem('auth_token');
  localStorage.removeItem('user_name');
  this.loggedIn = false;
  this._authNavStatusSource.next(false);
 }

 isLoggedIn() {
  return this.loggedIn;
 }
}
```

ConfigService.ts

```
import { Injectable } from '@angular/core';

@Injectable()
export class ConfigService {

  _apiURI : string;

  constructor() {
     this._apiURI = 'https://diplom20200410104007.azurewebsites.net/api';
   }

   getApiURI() {
      return this._apiURI;
   }
}
```